N° : 860 Année 2001

UNIVERSITÉ DE FRANCHE-COMTÉ

THÈSE

présentée a

L'UFR des Sciences et Techniques de l'Université de Franche-Comté

pour obtenir le

Grade de Docteur de l'Université de Franche-Comté
Spécialité Automatique et Informatique

Détermination systématique des graphes de précédence et équilibrage des lignes d'assemblage

par

Antoneta BRATCU

Soutenue le 10 juillet 2001 devant la Commission d'Examen :

G. JUANOLE	Professeur à l'Université de Toulouse	Président
J.-P. BOURRIÈRES	Professeur à l'Université de Bordeaux 1	Rapporteur
B. DESCOTES-GENON	Professeur à l'Université Joseph Fourrier, Grenoble	Rapporteur
A. BOURJAULT	Professeur à l'ENSMM, Besançon	Examinateur
A. El MOUDNI	Professeur à l'UTBM, Belfort	Examinateur
G. VALLET	Ingénieur à l'ADEPA	Examinateur
J.-M. HENRIOUD	Professeur à l'Université de Franche-Comté, Besançon	Directeur de thèse
V. MÎNZU	Professeur à l'Université de Galați, Roumanie	Directeur de thèse

À tous mes vrais amis…

"*Eu nu sunt altceva decât*
o pată de sânge
care vorbeşte."

"*Je ne suis*
qu'une tache de sang
qui parle."

Nichita STĂNESCU
Autoportret

Avant propos

Les résultats présentés dans cet ouvrage sont le fruit de recherches effectuées au sein de l'équipe "Méthodologie de l'Assemblage" du Laboratoire d'Automatique de Besançon (UMR CNRS 6596) ; ils ont été placés sous la direction scientifique de Monsieur Jean-Michel HENRIOUD, Professeur à l'Université de Franche-Comté et Directeur de l'École Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques.

Je tiens en tout premier lieu à lui exprimer ici ma gratitude pour m'avoir permis d'effectuer ces travaux ; je lui sais également gré des nombreux conseils qu'il n'a cessé de prodiguer tout au long de mes recherches.

Je voudrais aussi exprimer toute ma reconnaissance à Monsieur Alain BOURJAULT, Professeur à l'École Nationale Supérieure de Mécanique et des Microtechniques et Directeur du Laboratoire d'Automatique de Besançon, pour l'accueil qu'il m'a réservé dans son laboratoire, pour la chance qu'il m'a offert d'effectuer ces recherches et pour avoir accepté de participer au jury de cette thèse.

Je remercie Monsieur Jean-Paul BOURRIÈRES, Professeur à l'Université de Bordeaux 1, pour l'honneur qu'il m'a fait en acceptant d'être rapporteur de ma thèse.

Je tiens également à remercier Monsieur Bernard DESCOTES-GENON, Professeur à l'Université Joseph Fourrier de Grenoble, qui a lui aussi accepté la charge de rapporteur de cette thèse et a bien voulu participer à son jury.

Pour les mêmes raisons, je remercie Monsieur Guy JUANOLE, Professeur à l'Université de Toulouse, d'avoir accepté de juger ce travail et de présider le jury de la soutenance.

J'exprime aussi toute ma reconnaissance à Monsieur Abdellah EL MOUDNI, Professeur à l'Université Technologique de Belfort-Montbéliard, pour avoir bien voulu accepter de juger ce travail.

Je remercie également Monsieur Gérard VALLET, Ingénieur à l'Agence Nationale pour le Développement de la Productique Appliquée à l'Industrie (ADEPA), qui a lui aussi accepté de juger ma thèse.

Je tiens tout particulièrement à remercier Monsieur Viorel MÎNZU, Professeur et Doyen de la Faculté de Génie Électrique de Galați (Roumanie), ancien membre du L.A.B., pour son précieux soutien scientifique, ainsi que pour la qualité extrême du soutien moral qu'il m'a apporté.

J'adresse également ma reconnaissance à mes collègues compatriotes – Daniela CERNEGA, Sergiu CARAMAN, Laurențiu FRANGU, Dan DULMAN et Iulian MUNTEANU – qui, de près ou de loin, ont eu la patience de me supporter, conseiller et encourager. Qu'ils sachent que je leur suis redevable pour l'enthousiasme qui ne m'a pas quitté jusqu'à l'aboutissement de ce travail.

Je veux remercier plus spécialement mes vieilles amies Raluca SAHLEAN et Anca PUPĂZĂ pour leur bonne humeur et pour m'avoir soutenu dans quelques moments difficiles, parfois même pénibles.

Il est vraiment très difficile de remercier tous les collègues et amis qui ont su maintenir mon bon état d'esprit tout au cours de l'élaboration de cette thèse. Que chacun d'eux trouve ici l'expression de la plus profonde reconnaissance et de l'amitié que j'éprouve à leur égard.

En fin, mes pensées se dirigent vers ma famille et, surtout, vers mes parents, dont la dévotion et compréhension m'ont soutenu aux carrefours de ma vie. Je leur suis reconnaissante pour m'avoir appris la passion de bien faire les choses.

Table des matières

Introduction

Les réalités des marchés économiques imposent aux entreprises manufacturières des contraintes de plus en plus difficiles à remplir comme, par exemple : la diversification des produits offerts, l'amélioration de leur qualité, la diminution des coûts et des délais. Ces contraintes sont satisfaites non seulement par des sauts technologiques, mais aussi par une meilleure organisation des systèmes de fabrication, en utilisant les ressources techniques existantes. En fait, c'est cela la tâche de la productique, qui doit fournir des modèles et des méthodes pour structurer convenablement les systèmes de fabrication.

Les recherches qui font l'objet de cette thèse ont été menées dans le cadre de l'équipe "Méthodologie de l'assemblage" du Laboratoire d'Automatique de Besançon. Pour l'assemblage, cas particulier de la fabrication, qui bénéficie en même temps d'une certaine généralité, dans le cadre de cette équipe un certain nombre de thèmes de recherche ont été abordés au fil des années : élaboration des gammes d'assemblage, conception des systèmes flexibles d'assemblage, pilotage réactif, applications de l'ingénierie concurrente, modélisation et méthodes d'analyse des systèmes d'assemblage.

Le travail présenté dans ce rapport se place dans la problématique de la conception des systèmes flexibles d'assemblage. Ce domaine ne dispose pas d'un paradigme largement reconnu dans le monde scientifique ou industriel. Dans le temps, des méthodes de conception des systèmes d'assemblage se sont imposées grâce au compromis réalisé entre les différentes méthodes de modélisation et d'optimisation, d'une part, et les réalités et les habitudes industrielles, d'autre part.

Nos recherches n'ont pas eu comme objectif l'élaboration d'une nouvelle méthode de conception des systèmes d'assemblage, mais l'étude de méthodes et d'outils complémentaires aux approches de conception largement acceptées : méthodes C.S.D.L., méthode L.A.B., etc. Celles-ci utilisent dans les étapes de début de la conception des modèles simples ou empiriques des processus d'assemblage : séquences (gammes) d'assemblage, arbres ou graphes d'assemblage, graphes de précédence. L'utilisation de tels modèles est incontournable, puisque dans ces étapes on ne connaît pas les détails sur l'organisation du système d'assemblage. Une fois ce système conçu, on disposera d'éléments suffisants pour affiner les modèles et faire des simulations et des analyses.

Le graphe de précédence est un modèle empirique des processus d'assemblage, mais qui est largement utilisé dans les méthodes de conception susmentionnées. Malgré l'utilisation préférentielle du graphe de précédence par ces méthodes, elles ne fournissent pas de

procédures d'obtention d'un tel graphe. Cette tâche est confiée au concepteur, qui doit utiliser son expérience et faire une analyse plus ou moins subjective du produit à assembler. En revanche, pour les séquences ou les arbres d'assemblage on dispose de méthodes et d'algorithmes vérifiés de génération.

C'est pour cela que notre démarche de recherche a eu comme but l'utilisation de ces résultats, afin d'élaborer une procédure systématique de construction des graphes de précédence, en vue de leur utilisation comme donnée d'entrée pour les méthodes d'équilibrage des systèmes d'assemblage.

Notre travail s'articule autour de cinq chapitres, de la façon décrite ci-après.

Le premier chapitre est dédié à la description de la problématique de la conception des systèmes d'assemblage. Nous avons passé en revue les différentes approches sur ce sujet, conjointement avec les modèles des processus d'assemblage utilisés par chacune d'elles.

Suite à la mise en évidence de la supériorité des graphes de précédence comme modèle des processus d'assemblage, nous présentons au deuxième chapitre les propriétés de ceux-ci par rapport à d'autres modèles : séquences (gammes) d'assemblage, arbres d'assemblage, graphes d'assemblage.

Nous présentons également un état de l'art des approches de génération des graphes de précédence.

Nous énonçons d'une manière informelle l'objectif d'élaboration d'une méthode systématique d'obtention des graphes de précédence à partir de l'ensemble de gammes d'assemblage d'un produit donné.

Le troisième chapitre est entièrement consacré à la résolution mathématique de ce problème.

Nous démontrons une condition nécessaire et suffisante pour l'équivalence d'un ensemble de gammes avec un et un seul graphe de précédence. Cette condition repose sur une propriété globale de l'ensemble de gammes, mise en évidence à l'aide de la relation d'indifférence – définie sur l'ensemble des nœuds du graphe, comme complémentaire à la relation de précédence – et du graphe correspondant, appelé graphe d'indifférence.

Pour le cas général, où la condition susmentionnée n'est pas remplie, nous présentons un algorithme de détermination de l'ensemble des graphes de précédence équivalent à un ensemble donné de gammes.

Dans le quatrième chapitre nous présentons les méthodes de conception des systèmes d'assemblage issues de l'équilibrage des lignes, qui utilisent comme modèle des processus d'assemblage le graphe de précédence, aussi bien dans le cas monoproduit, que dans le cas multiproduit. Nous donnons la formulation du problème d'équilibrage en tant que problème d'affectation optimale des tâches aux postes de travail. Le critère d'optimalité est la minimisation du temps de cycle total du système.

Nous passons en revue différentes méthodes d'optimisation pour l'équilibrage. Nous donnons une formulation systémique du problème d'équilibrage, en tant que problème d'optimisation discrète, adaptée à la résolution par programmation dynamique.

Le dernier chapitre est dédié à l'étude d'une classe spéciale de systèmes d'assemblage : les lignes d'assemblage avec auto-équilibrage. Ces systèmes, qui utilisent des opérateurs humains, peuvent spontanément, sans intervention consciente, se comporter de façon optimale du point de vue de l'équilibrage, à condition qu'ils remplissent une certaine contrainte de placement des ouvriers sur la ligne. Grâce à cette propriété, de telles lignes présentent l'avantage de pouvoir être conçues sans besoin de résoudre un problème d'ALB classique, lorsque le temps de cycle total est implicitement minimisé.

À partir d'un modèle construit sous quelques hypothèses simplificatrices – connu dans la littérature sous le nom de "modèle normatif" – nous proposons d'abord une analyse par simulation, puis une analyse systémique originale de ce type de système, en utilisant le formalisme des systèmes dynamiques hybrides. Ce sont des systèmes dont la dynamique est aussi bien continue, que due à d'événements discrets.

En utilisant les critères de stabilité des systèmes dynamiques discrets, nous donnons une autre démonstration de la condition suffisante qui assure l'auto-équilibrage du système.

Une conclusion générale est organisée à la fin pour fournir une vision globale sur notre travail.

L'annexe A est consacrée aux démonstrations des principaux résultats théoriques obtenus à partir du modèle normatif des lignes d'assemblage avec auto-équilibrage.

Dans l'annexe B nous présentons le schéma-bloc non linéaire implémenté en Simulink, utilisé à la simulation des lignes avec auto-équilibrage.

Les principaux éléments du formalisme des systèmes dynamiques hybrides à mouvements discontinus sont détaillés dans l'annexe C.

L'annexe D regroupe les énoncés de quelques critères de stabilité des systèmes dynamiques discrets.

Pour faciliter la lecture, nous établissons à la fin de ce mémoire un répertoire de figures et un index alphabétique.

I PROBLÉMATIQUE DE L'ASSEMBLAGE

I.1 Introduction

I.1.1 Concepts de base. Spécificité de l'assemblage

L'assemblage est l'action de fabriquer des objets par l'agrégation d'objets plus simples. Un système d'assemblage a la structure générale présentée dans la figure I-1 [HENR 89], comportant p flux d'entrée, Φ_i^e, $i=1,2,...p$, et q flux de sortie, Φ_j^s, $j=1,2,...q$.

Fig. I-1. Schéma de principe d'un système d'assemblage

Les flux d'entrée portent sur des objets c_i tous identiques, appelés **composants élémentaires**. Les flux de sortie portent sur des objets P_j également tous identiques, appelés **produits finis**. Ces appellations sont relatives au système considéré : certains composants élémentaires peuvent être des produits finis pour d'autres systèmes placés en amont, tandis que certains produits finis jouent le rôle des composants élémentaires pour des systèmes placés en aval.

Un système d'assemblage avec un seul flux de sortie – $q=1$ – s'appelle **système monoproduit**. Si $q>1$, on parle d'un **système multiproduit**. Dans ce dernier cas, q désigne le nombre de types de produits sortis, qui forment une **famille de produits**.

L'ensemble des composants élémentaires nécessaires à la fabrication par assemblage d'un exemplaire d'un objet p_j constitue le **pré-produit** de P_j. Les objets qui existent à un instant donné et qui sont issus des transformations successives du pré-produit afin de fabriquer le produit fini représentent le **produit intermédiaire**. Chacune de ces transformations élémentaires est appelée **opération**. Le terme "opération" ne regarde pas uniquement l'assemblage – c'est à dire, l'agrégation proprement dite d'objets – mais aussi d'autres opérations, comme, par exemple, le marquage, le contrôle, l'emballage, etc. Chaque objet particulier, élément du produit intermédiaire à un instant donné, est appelé **constituant**.

L'ensemble des opérations concernant la production d'un flux de sortie est appelé **processus d'assemblage**. Le temps est implicitement contenu dans la notion de "processus", mais il est réduit à une chronologie partiellement décrite par les contraintes de précédence entre opérations.

Selon sa définition, le système d'assemblage désigne en mode générique toute réalité matérielle comportant en entrée au moins deux composants élémentaires et en sortie un produit fini. Le système peut être représenté aussi bien par une simple machine que par un atelier, s'il répond à la définition ci-dessus.

Du point de vue de l'automatisation, l'assemblage présente des désavantages par rapport à l'usinage. Cela s'explique par certains ***caractères spécifiques*** :

- *caractère multi-variable*
  Les systèmes d'assemblage sont des systèmes multivariables, au moins multi-entrées. Nous pouvons nous imaginer la structure interne d'un système d'assemblage comme un arbre dont les arcs sont parcourus par des *flux d'objets rigoureusement synchronisés*. En chaque nœud, est assurée l'exactitude du rendez-vous des constituants qui doivent y être assemblés. Cette contrainte de rendez-vous ne se rencontre pas en usinage.
- *multiplicité des processus d'assemblage*
  Il existe en général plusieurs manières pour assembler un seul type de produit. Chacune d'elles représente un processus d'assemblage (gamme) admissible, caractérisé par une circulation particulière des flux d'objets, qui engendre une structure particulière du système d'assemblage. Ces systèmes possèdent des performances différentes. Le choix des processus d'assemblage est un problème difficile lors de la conception d'un système d'assemblage.
- *durée d'une opération d'assemblage*
  En assemblage les temps opératoires sont beaucoup plus courts qu'en usinage, ce qui rend comparativement pénalisants les temps de transport. De cette façon, le choix et l'implantation des systèmes de transfert acquirent plus d'importance qu'en usinage.
- *hétérogénéité des équipements*
  Il n'existe pas de machines universelles d'assemblage comme il existe, par exemple, des centres d'usinage. Pour chaque processus d'assemblage il faut créer des machines spécifiques et/ou combiner un nombre important de machines existantes et de manipulateurs.
- *présence fréquente d'aléas*
  La diversité des composants impliqués et des opérations à effectuer sur chaque exemplaire, même d'un seul produit, ainsi que la multitude des équipements composant le système d'assemblage impliquent une fréquence élevée des divers incidents (défauts d'alimentation, défauts de réalisation d'une opération, etc.). Les aléas réduisent la productivité et/ou la qualité de l'assemblage, mais ils peuvent être contrecarrés par des politiques appropriées concernant l'organisation du système d'assemblage.
- *aspect multiproduit*
  La conception d'un système d'assemblage multiproduit est un problème nouveau par rapport à l'usinage. Dans l'assemblage multiproduit on a besoin d'un modèle de la famille de produits assemblés – qui doit mettre en évidence les propriétés globales de celle-ci – ce qui n'est pas le cas en usinage.

I.1.2 Analyse des systèmes d'assemblage

I.1.2.1 Les équipements d'assemblage

Il existe deux catégories principales d'équipements qui participent à la réalisation des exemplaires du produit dans un système d'assemblage (voir figure I-2) :

- les *opérateurs*, qui principalement réalisent effectivement les opérations d'assemblage constitutives, mais ils peuvent assumer aussi des opérations adjacentes aux opérations constitutives (chargement, déchargement, transport, etc.) ;
- les *équipements auxiliaires*, qui assistent et facilitent l'exécution des opérations par les opérateurs.

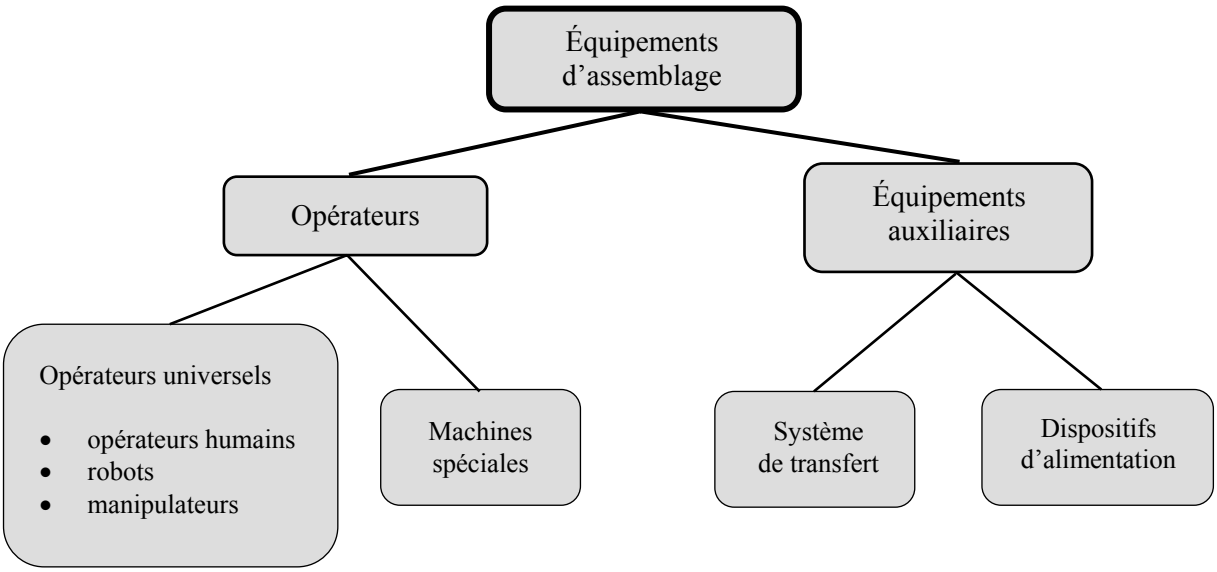


Fig. I-2. Classification des équipements d'assemblage

Selon la nature des opérations réalisées, il existe deux sortes d'opérateurs :

- les *opérateurs universels*, qui sont en principe capables d'effectuer la plupart des opérations d'assemblage par l'intermédiaire de différents outils ;
- les *machines spéciales*, qui sont destinées à un nombre réduit d'opérations, généralement irréalisables par les opérateurs universels.

Les équipements auxiliaires ne contribuent pas directement à la constitution du produit fini, comme dans le cas des opérateurs. Il est possible de distinguer deux types de tels équipements :

- *le système de transfert*, dont la fonction principale est d'assurer la circulation des constituants primaires ; il réalise aussi les fonctions d'indexage et de gestion des fils d'attente ;
- *les dispositifs d'alimentation*, destinés à fournir au système d'assemblage des composants élémentaires ou certains outils ; dans ce groupe on trouve : les bols vibrants, les chariots, les stocks de composants élémentaires, etc.

Le système de transfert se présente souvent sous la forme d'un réseaux de circulation des constituants du produit fini. Pour des raisons de commandabilité et de productivité, un tel réseau doit être facilement conçu, ce qui a conduit au développement d'unités de transfert modulaires et ensuite à la prédominance d'*architectures types* au sein des entreprises manufacturières. Ainsi, la plupart des réalisations est couverte par les architectures *en ligne*, *en arbre* ou *en anneau*.

I.1.2.2 Structure des systèmes d'assemblage

Dans la thèse de R. Olivier [OLIV 86] on trouve une analyse des systèmes d'assemblage monoproduits, basée sur une approche orientée produit. Son idée de base est de faire intervenir les actions qui modifient l'état du produit assemblé : les transports et les traitements appliqués aux objets.

La structure générale des systèmes d'assemblage est obtenue par une technique de *décomposition itérative*. Un système d'assemblage est formé par plusieurs sous-systèmes connectés par des actions de transport. Ces sous-systèmes sont définis par les traitements auxquels le produit est soumis ; ils restent valables y compris pour les systèmes d'assemblage multiproduits.

La démarche d'analyse par décomposition utilise des *critères fonctionnels* pour définir les sous-systèmes, permettant d'introduire les concepts de base dans la structuration d'un système d'assemblage comme, par exemple, "atelier", "îlot", "cellule", "poste", "opération", "temps de cycle", "temps de traitement", etc.

Nous présentons ensuite les grandes lignes de cette démarche.

On part de l'encadrement de l'assemblage dans l'activité de production d'une entreprise, qui se déroule dans le **centre de production**. Le terme de "centre de production" désigne l'ensemble des moyens assurant l'activité de **production** de l'entreprise ; il s'oppose ainsi à l'ensemble des autres services traitant de l'information [BOUJ 90]. L'essentiel du centre de production est la circulation des *flux matériels*.

Un centre de production peut être décomposé en **départements** – approvisionnement, outillage, maintenance, expédition, fabrication, qualité, etc. – dont nous intéresse particulièrement le **département fabrication**, qui a pour rôle d'élaborer les produits à l'aide de diverses technologies : usinage, moulage, traitements thermiques, assemblage, etc. En correspondance, le département fabrication admet une décomposition en **ateliers technologiques**. Nous nous concentrons sur l'atelier d'assemblage.

La fonction de l'**atelier d'assemblage** est de traiter des composants pour obtenir des produits finis. Comme nous l'avons mentionné auparavant, les opérations d'assemblage ne se réfèrent pas seulement à l'assemblage. Les frontières, parfois matérialisées, de cet atelier sont traversées par des flux matériels de sortie, aussi bien que par des flux matériels d'entrée, qui s'intègrent au processus d'assemblage.

En utilisant ensuite la technique de décomposition, un atelier d'assemblage est considéré comme un ensemble d'**îlots**. Ceux-ci peuvent être caractérisés soit par la même technologie, soit par la fabrication d'un même produit ou d'une famille de produits. Quelque soit le cas, la définition d'un îlot reste la même.

Un **îlot d'assemblage** est un sous-ensemble de ressources de l'atelier d'assemblage, assurant la fabrication complète d'un produit déterminé P par l'assemblage d'un ensemble de m composants élémentaires. La démarche de décomposition étant orientée produit, la caractérisation d'un îlot d'assemblage se fait sur un horizon temporel qui correspond à un nombre de produits finis significatif du point de vue de la gestion de ce niveau (jours, semaine,…). La durée des campagnes de fabrication du produit détermine la dimension de cet horizon.

L'îlot d'assemblage possède une frontière réelle, parfois matérialisée, traversée par des flux d'objets de deux types : flux d'entrée et flux de sortie. Les entrées peuvent être de l'intérieur ou de l'extérieur de l'entreprise, ou bien du recyclage. Les sorties sont constituées des produits finis, des rebuts et du recyclage d'outils. Dans la thèse de V. Mînzu [MÎNZ 95] on trouve une description plus détaillée de ces flux.

Un problème spécifique de l'assemblage est la résolution des rendez-vous des constituants impliqués dans une opération d'assemblage. Outre les difficultés de pilotage, un tel problème engendre des *problèmes de fiabilité*. La fiabilité d'un ensemble d'équipements connectés en série est donnée par le produit des coefficients de fiabilité de chacun des équipements. D'où émerge la recommandation de ne pas enchaîner plus de six à douze opérations du processus d'assemblage.

Compte tenu de cet aspect, un îlot d'assemblage est conçu comme un ensemble de plusieurs groupes autonomes d'équipements connectés par des stocks tampons. Ces groupes, constituant des sous-systèmes de l'îlot d'assemblage, s'appellent **cellules d'assemblage**.

Une **cellule d'assemblage** est une partie d'un îlot d'assemblage :

- capable de fonctionner de manière autonome et satisfaisante pendant une durée limitée ;
- non sécable en unités plus petites répondant à la propriété précédente.

L'essentiel du concept de "cellule d'assemblage" est :

- le fonctionnement autonome à court terme ;
- la présence des stocks tampons.

Du point de vue du processus d'assemblage, une cellule doit fabriquer par assemblage un produit intermédiaire PI, à partir de p constituants, qui sont :

- soit des composants élémentaires ;
- soit des objets assemblés par d'autres cellules (*sous-assemblages*).

Notons que $p \leq m$, où l'égalité caractérise la situation où l'îlot contient une seule cellule.

Le comportement dynamique d'une cellule d'assemblage peut être décrit par deux valeurs :

- **le temps de cycle** (T_c), qui est la durée normale séparant les apparitions successives de deux produits intermédiaires PI à la sortie ;
- **le temps de traitement** (T_t), c'est à dire la durée pendant laquelle une instance du produit subit le traitement caractéristique de la cellule.

Les deux valeurs sont liées par la relation :

$$T_t = \mathrm{NI} \cdot T_c,$$

où la valeur NI s'appelle **en-cours** et représente le nombre d'exemplaires du produit en cours de fabrication à l'intérieur de la cellule, à un moment donné.

Nous complétons la description du comportement dynamique de la cellule par la notion de "**en-cours de stockage**". Cela représente le nombre d'exemplaires de produits intermédiaires qui peuvent être assemblés dès qu'on supprime l'alimentation des stocks de la cellule. Cette valeur est une mésure de l'*inertie* de la cellule face à une reconfiguration requise par un nouveau lot de fabrication.

La connexion de la cellule à son environnement est réalisée par deux types de flux matériels :

- des flux *entre* cellules ;
- des flux facultatifs assurant *la liaison avec l'extérieur* de l'îlot ; ils peuvent être :
  - unitaires
  - ou collectifs.

À l'intérieur d'une cellule d'assemblage, il y a plusieurs exemplaires du produit en cours d'assemblage, mais dans des phases différentes. Dans [OLIV 86] on trouve une description claire pour cette situation :

"Chaque exemplaire subit un certain nombre de transformations (ex. : adjonction de composants) sur une partie donnée de l'équipement de la cellule, avant d'être transféré sur

une autre partie, pour y subir de nouveaux traitements, tandis qu'un autre exemplaire prend la place ainsi libérée."

Le sous-système de la cellule d'assemblage mis en évidence ci-dessus s'appelle **poste d'assemblage**. Il regroupe les ressources nécessaires à la fabrication d'un produit intermédiaire donné PI'. Le caractère dynamique du poste d'assemblage est décrit par **le temps de cycle du poste**, T_p, c'est à dire la durée du séjour d'un seul exemplaire du produit à assembler à l'intérieur du poste.

Donc, la sortie d'un poste d'assemblage est un produit intermédiaire PI', obtenu à partir de q constituants, où $q \leq p \leq m$ (rappelons que p est le nombre de constituants en entrée de la cellule et m est le nombre de constituants en entrée de l'îlot). Si la cellule comporte un seul poste, alors $q=p$. Si $q=m$, l'îlot, la cellule et le poste se confondent.

Remarquons que l'en cours du poste est unitaire.

À l'intérieur d'une cellule, les postes sont souvent rangés selon un ordre total. Pourtant, cet ordre de rangement peut également être partiel, permettant le *parallélisme* entre les postes. Cela se passe, par exemple, quand les produits intermédiaires de deux postes seront des constituants pour la même instance du produit fini.

Pour des raisons de productivité, certains systèmes d'assemblage pratiquent la multiplication des postes. Un **poste multiple** contient plusieurs postes identiques, produisant des constituants identiques et travaillant en parallèle.

Remarquons la différence entre le traitement des constituants en avant d'une cellule et dans ses stocks – traitement *collectif* – et le traitement *individuel* appliqué par chaque poste de travail.

Dans la thèse [MÎNZ 95] on trouve un développement du concept de "poste d'assemblage", surtout du point de vue constructif.

La fonction d'un poste d'assemblage est de faire progresser la fabrication du produit, en apportant au produit intermédiaire un ou plusieurs des caractères exigés par le produit fini, appelés *caractères fonctionnels*. Cela se fait généralement par l'intermédiaire d'un certain nombre d'**opérations**. Les caractères fonctionnels sont conférés par les **opérations fonctionnelles**.

Une opération s'effectue sur un très petit nombre, r, de constituants d'entrée pour réaliser un produit intermédiaire déterminé PI", où $r \leq q \leq p \leq m$ (nous avons conservé les notations antérieures). Le plus fréquemment, il s'agit d'opérations binaires (r=2) et d'opérations unaires (r=1). La durée d'une opération, T_r, est évidemment inférieure au temps de cycle du poste dont elle fait partie :

$$T_r \leq T_p$$

Au sein d'un poste d'assemblage, il est parfois nécessaire d'exécuter des déplacements courts entre deux opérations, comme, par exemple, des réorientations demandées par les contraintes spatiales de l'opération suivante.

I.1.3 Conception des systèmes d'assemblage

Tout produit est défini par les fonctions qu'il aura à remplir, par le niveau de performance auquel il devra les accomplir et par un ensemble de diverses contraintes. À la définition complète du produit on intègre certains caractères destinés à faciliter son assemblage. La définition de ces éléments fait l'objet de la *modélisation des produits*. On peut demander la reconception d'un produit qui s'avère trop coûteux à assembler.

Ensuite, il faut concevoir le système d'assemblage pour assembler le produit donné. Ce système est soumis à un ensemble de contraintes dont la principale est imposée par les *processus d'assemblage admissibles*, puisqu'ils montrent la succession des transformations

subies par tout exemplaire de produit en cours d'assemblage vers le produit fini. À la base de toute méthode de conception des systèmes d'assemblage se trouve, donc, la **représentation des processus d'assemblage**, dont la qualité influence beaucoup la performance du système conçu. Les plus connues méthodes de modélisation des processus d'assemblage seront présentées dans le paragraphe suivant, conjointement aux méthodes de conception qui les utilisent.

Dans la littérature il existe de nombreuses démarches de modélisation des processus d'assemblage. La qualité d'une représentation ou d'une autre dépend aussi bien de sa définition proprement dite, que de la facilité de générer les processus d'assemblage.

Une série de travaux de recherche menés au sein du Laboratoire d'Automatique de Besançon [BOUJ 84][HENR 89] a permis de proposer **les arbres d'assemblage** en tant que représentation des processus d'assemblage. Un logiciel pour générer ces arbres a été également mis au point. Ce type de modèle est susceptible d'être critiqué quant à son utilisation pour la conception des systèmes d'assemblage, mais il s'est en avéré un des meilleurs.

Dans le paragraphe suivant nous présentons un état de l'art des méthodes de conception des systèmes d'assemblage. Les plus connues et les plus répandues dans la pratique sont celles qui utilisent comme représentation des processus d'assemblage le **graphe de précédence**, ayant comme objectif l'*équilibrage des lignes d'assemblage*. Elles sont nommées **méthodes d'ALB**, selon leur appellation anglaise Assembly Line Balancing. Ce problème regarde la structuration optimale d'un système d'assemblage en postes de travail, y compris le choix des équipements et l'affectations des opérations. Le résultat est le meilleur équilibrage des postes du point de vue des temps de travail.

Comme outil mathématique général, l'utilisation du graphe de précédence n'est pas récente. Par exemple, dans les problèmes d'ordonnancement on utilise souvent le graphe PERT, qui est toujours un graphe de précédence. En assemblage, le graphe de précédence décrit directement les contraintes d'antériorité entre opérations (ou *tâches* – cf. I.2.2.1) d'assemblage, étant l'image de la *relation d'ordre partiel* sur l'ensemble de celles-ci.

Dans la thèse [MÎNZ 95] on montre que les méthodes d'ALB reposent sur une propriété de structure pour les postes d'assemblage, par rapport au graphe de précédence (idée reprise dans le chapitre IV de ce travail).

Mais la grande difficulté éprouvée par ces méthodes est l'**obtention des graphes de précédence**. Les méthodes traditionnelles, qui présupposent l'unicité des graphes de précédence, sont convenables pour la reconception des lignes existantes ou pour des produits simples. Quant à la conception d'une nouvelle ligne, le problème est assez différent, lorsqu'il faut trouver l'ensemble de tous les graphes de précédence qui sont jugés bons. Chacun d'eux constitue une donnée d'entrée pour une méthode d'ALB, et le système optimal résulte de l'évaluation des différentes solutions.

Une des origines de la difficulté de l'obtention des graphes de précédence se trouve dans le *passage direct* du modèle du produit aux graphes de précédence, méthode proposée par quelques travaux de recherche [KO 87][FROM 88][LEE 88][WEUL 89][DELC 90b].

La méthode indirecte de génération des graphes de précédence développée dans la thèse [CHEN 96] utilise comme point de départ les arbres d'assemblage. Cette méthode combine deux approches existantes : la méthode L.A.B. d'obtention des arbres d'assemblage, qui sont directement utilisables pour la conception des systèmes d'assemblage, et les méthodes d'ALB qui exploitent les graphes de précédence obtenus de façon empirique. En fait, la génération des graphes de précédence se fait à partir de l'ensemble d'opérations qui composent les arbres d'assemblage élaborés, sous deux hypothèses que cet ensemble doit remplir.

La résolution de ce problème est basée sur le *principe de recherche heuristique*, qui la rend assez compliquée. Il est prévisible que la complexité de la méthode est forte ; néanmoins, elle n'a pas été estimée. Même s'il existe des propositions d'amélioration, on n'a pas encore réalisé un prototype de logiciel afin de tester l'efficacité et le besoin de calcul de la méthode proposée.

I.1.4 Objectif du travail

Notre travail présenté ici est organisé en deux grandes parties. La première partie est dédiée à l'étude des graphes de précédence pour l'assemblage, ayant comme objectif une nouvelle méthode d'obtention de ceux-ci. La deuxième partie regarde l'utilisation des graphes de précédence par les méthodes d'ALB de conception des systèmes d'assemblage. Nous nous intéressons particulièrement aux systèmes d'assemblage dont l'équilibrage émerge spontanément de leur structuration spécifique.

Dans la première partie nous proposons une méthode systématique de génération des **graphes de précédence** à partir de l'**ensemble des gammes d'assemblage** d'un produit donné. Notre méthode s'appuie sur l'existence d'une propriété structurelle de l'ensemble des gammes, qui permet sa représentation équivalente par *un seul* graphe de précédence. En fin du développement de notre démarche, nous proposons un algorithme qui génère l'ensemble des graphes de précédence correspondant à un ensemble donné de gammes dans le cas général, où la propriété susmentionnée n'est pas remplie.

En tant que modèle des processus d'assemblage, les graphes de précédence constituent la donnée d'entrée pour la conception des systèmes d'assemblage optimaux du point de vue du temps de travail. Dans la deuxième partie de ce travail nous présentons le problème de l'équilibrage des systèmes d'assemblage, généralement formulé comme un problème d'optimisation du temps de cycle.

Les **systèmes d'assemblage avec auto-équilibrage** forment une classe spéciale. Il est suffisant que leur structuration interne remplisse une certaine condition, pour qu'ils se comportent de manière optimale. Donc, l'équilibrage de ces systèmes est une propriété intrinsèque. Nous développons une **analyse systémique** originale de ce type de systèmes d'assemblage à l'aide de la théorie des systèmes dynamiques hybrides. Nous donnons une nouvelle démonstration de la condition suffisante assurant le fonctionnement optimal.

I.2 Méthodes de conception des systèmes d'assemblage

I.2.1 Cadre général de la conception des systèmes d'assemblage

Le modèle général de la conception des systèmes d'assemblage comporte deux parties :

- les **contraintes** exprimant la spécificité du problème formulé ;
- les **critères d'optimisation**.

La définition des éléments du modèle ci-dessus repose sur la structure générale d'organisation des systèmes d'assemblage, laquelle résulte de l'analyse par décomposition (voir I.1.2). Cette structure hiérarchique décrivant un système d'assemblage en cours d'exploitation sera détaillée pour chaque méthode de conception.

Remarque :

On admet en général qu'une **tâche d'assemblage** représente la contribution d'un opérateur à la réalisation d'une opération. Une tâche englobe en général plusieurs opérations.

Dans la définition du graphe de précédence les significations des deux notions se confondent (voir I.2.2.1).

Compte tenu de la signification générale de la notion de "tâche d'assemblage", à l'étape de la conception, la structure d'un système d'assemblage est principalement déterminée par deux éléments :

- l'organisation des opérateurs en **postes de travail**
- et l'**affectation des tâches** d'assemblage à chaque poste.

Ensuite, la mise en œuvre du système de transfert et le choix des équipements de chaque poste sont techniquement limités.

En conclusion, toute méthode de conception doit fournir les données suivantes :

- nombre de postes d'assemblage ;
- type d'opérateur devant être installé au sein de chaque poste ;
- types d'outils devant équiper chaque opérateur ;
- tâches d'assemblage affectées à chaque poste.

I.2.1.1 Contraintes agissant sur la conception

Essentiellement, les contraintes qui agissent sur la conception des systèmes d'assemblage regardent les ressources du système en général. Il s'agit des ressources matérielles et du temps, qui peut être également considéré comme une ressource.

Contraintes d'antériorité entre tâches d'assemblage

Ces contraintes découlent des processus d'assemblage, lorsque les tâches qui doivent être affectées aux postes de travail sont soumises aux contraintes d'antériorité. Nous remarquons que trois types de modèles des processus d'assemblage sont les plus adaptés à représenter ce type de contraintes :

- les *séquences (gammes) d'assemblage* ;
- les *arbres d'assemblage* ;
- les *graphes de précédence*.

Pour cette raison, les modèles ci-dessus sont les plus souvent utilisés par les méthodes de conception, même s'ils ne tiennent pas nécessairement compte de quelques aspects spécifiques, comme :

- les changements d'outils,
- la réorientation des constituants avant une opération,
- le temps masqué dû aux équipements qui travaillent en parallèle,
- le lancement de production et les changements de série.

La possibilité de tenir compte de ces problèmes n'est pas obligatoire. Néanmoins, elle constitue un critère d'appréciation de toute méthode de conception. Donc, leur prise en compte doit se faire soit de manière explicite dans la représentation des processus d'assemblage, soit implicitement dans l'algorithme appliqué.

Contraintes sur les équipements

Trois éléments décrivent ce type de contraintes :

- les types d'opérateurs accessibles,
- les outils disponibles,
- les temps d'exécution.

Chaque opération d'assemblage peut être réalisée par un ensemble d'opérateurs. Il en résulte plusieurs couples opération-opérateur – qui est, en fait, une tâche – chacun caractérisé par un temps d'exécution. À chaque type d'opérateur est donné un ensemble d'outils accessibles, dont le choix influence peu le temps d'exécution.

Dans l'assemblage manuel il est logique que les temps d'exécution subissent des variations stochastiques. Ce qui n'est pas le cas dans les systèmes d'assemblage automatisés, où le temps d'exécution d'un couple opération-opérateur est précisément déterminé.

Contrainte spatiale

À chaque poste d'assemblage on impose un seuil pour la superficie de l'espace qu'il occupe. Ainsi, la contrainte spatiale se traduit dans la requête que la somme des superficies des espaces allouées aux équipements d'un poste ne dépasse pas ce seuil.

Contrainte topologique

Cette contrainte est liée à l'architecture du système de transfert au sein d'un système d'assemblage. Dans quelques travaux de recherche [VALL 90][BARA 91] on propose des systèmes d'assemblage *réactifs*, basés sur des architectures *en anneau*, permettant à une même palette de passer plusieurs fois dans un même poste d'assemblage au cours de la réalisation d'un même exemplaire de produit fini.

Mais en général cette situation n'est pas recommandable, ce qui conduit à la formulation d'une contrainte topologique entre tâches d'assemblage : une tâche peut être affectée à un poste si et seulement si celles qui conditionnent sa réalisation sont déjà affectées à ce poste ou à ceux qui le précèdent.

Contrainte temporelle

La conception d'un système d'assemblage est soumise à un objectif de production prévu, qui est exprimé par un volume de production annuel et, finalement, par un temps de cycle maximal imposé au système (noté par T_c – voir I.1.2). Il en résulte une contrainte temporelle qui porte sur chaque poste d'assemblage : le temps de travail effectif du poste, qui est la somme des temps d'exécution des tâches affectées, ne doit pas dépasser T_c.

I.2.1.2 Critères d'optimisation

On distingue deux classes de critères d'optimisation utilisés dans la conception des systèmes d'assemblage : les critères techniques et les critères économiques.

La plupart des critères techniques s'apparentent aux méthodes de la recherche opérationnelle. La minimisation du nombre de postes d'assemblage a été le plus employé critère technique. Il conduit à une structure où les postes contiennent un maximum de tâches, de sorte qu'on assure en bonnes conditions l'objectif d'obtenir pour chaque poste un temps de travail inférieur ou égal au temps de cycle imposé.

Les critères économiques sont à prendre en compte lors de l'utilisation massive des équipements automatiques au détriment des opérateurs humains. Le plus souvent, les investissements initiaux dans ces équipements dépassent leurs coûts d'exploitation. C'est ainsi que la conception d'un système d'assemblage optimal du point de vue économique repose sur l'utilisation des modèles financiers essentiellement basés sur une distinction entre le coût d'investissement et le coût (variable) de fonctionnement associés à chaque équipement.

I.2.2 Méthodes systématiques

I.2.2.1 Méthodes d'ALB

Le problème de l'équilibrage des lignes d'assemblage a été premièrement formulé par Salveson en 1955. Dans les travaux [VALL 87], [ENME 89] et [GHOS 89] on trouve des états de l'art. Le problème d'ALB a été le point de départ d'une méthode globale de conception des systèmes d'assemblage à l'aide d'un ensemble de règles générales [WU 87]. [CHOW 90] constitue un ouvrage complet sur la conception des lignes d'assemblage par ALB.

À travers le temps, les nouveaux travaux de recherche ont généralement considéré deux types de modèles [BAYB 86] :

- le modèle *simple* (SALBP - Simple Assembly Line Balancing Problem)
- et le modèle *général* (GALBP – General Assembly Line Balancing Problem).

Le modèle simple est *déterministe* et *monoproduit*. Parmi ses hypothèses nous pouvons remarquer : l'existence des contraintes d'antériorité entre tâches, la faisabilité de toutes les tâches, l'insignifiance des coûts fixes et variables associés aux équipements, etc.

Par rapport au modèle simple, le modèle général est caractérisé par les données supplémentaires suivantes [BAYB 86] :

- production de plusieurs types de produits dans des séries différentes (*multi-model*),
- production de plusieurs types de produits dans une même série (*mixed-model*),
- restriction au groupement de certaines tâches dans un même poste d'assemblage (*zoning constraints*),
- existence potentielle des postes parallèles (*parallel stations*),
- possibilité de produire des sous-assemblages dans des sous-lignes d'assemblage (*parallel subassembly lines*).

Les méthodes de conception issues de l'équilibrage sont essentiellement orientées vers l'utilisation des *opérateurs humains* dans les systèmes d'assemblage. Par conséquent, le coût économique est réduit au seul nombre de postes et le coût d'investissement en équipements peut être ignoré. Nous montrons ensuite quelques aspects particuliers des deux éléments du modèle général de conception – les critères d'optimisation et les contraintes – dans le cas des méthodes d'ALB.

En ce qui concerne les critères techniques, le plus communément utilisé est *la minimisation du nombre de postes* d'assemblage. Comme il peut exister plusieurs solutions, quelques fois on utilise deux critères supplémentaires pour en choisir une :

- *minimisation du temps total d'inactivité* (de la somme des temps non occupés des postes) ;
- *maximisation du taux d'équilibrage de la ligne* (ce taux étant inversement proportionnel au temps d'inactivité).

Parmi les critères économiques employés dans la littérature d'ALB nous pouvons remarquer :

- minimisation du coût combiné des opérateurs humains, des postes d'assemblage et de la production inachevée ;
- minimisation du coût humain pour chaque exemplaire de produit ;
- minimisation du coût de pénalité dû aux temps improductifs, etc.

Le critère économique proposé par Peter Pinto [PINT 83] est la somme de trois coûts : le coût horaire des opérateurs humains, le surcoût dû aux heures supplémentaires et le coût d'investissement des équipements.

La plus importante des contraintes prises en compte par les méthodes d'ALB est la contrainte topologique (nommée aussi *contrainte de structure* [MÎNZ 95]). Elle est basée sur l'obtention des *postes "candidats"*. La définition de ceux-ci repose sur la définition des *structures de poste d'assemblage* : chaque poste candidat est une différence entre deux telles structures.

Ces définitions expriment une contrainte agissant sur toute procédure de découpage en postes, dont la formulation exacte sera présentée au paragraphe IV.2.2. Cette formulation se fait par rapport au graphe de précédence, dont l'étude est présentée aux chapitres II et III de ce travail. Nous nous contentons pour l'instant de rappeler qu'un graphe de précédence est formalisé par un couple (E, U) où :

- E est l'ensemble des nœuds représentant les tâches d'assemblage ;
- U est l'ensemble des arcs représentant les contraintes de précédence définies sur $E \times E$.

Remarque :

Le concept de *"tâche d'assemblage"*, tel qu'il est utilisé dans le contexte des graphes de précédence, est voisin de celui de *"opération"*, tel que nous l'avons défini plus haut.

Il en diffère toutefois par le fait que le constituant primaire associé à une "tâche" n'est que partiellement défini. En effet, sur l'exemple donné à la figure I-3, le composant primaire associé à la tâche *e*, par exemple, peut être (selon le processus mis en œuvre) :

abcd ou *abcdf* ou *abcdg* ou *abcdfg* ou *abcdgf*

Dans ce qui suit nous utiliserons la notion de "tâche" dans le sens de "opération".

Un sous-ensemble H de tâches est conforme à la *structure d'un poste d'assemblage* si toute tâche qui succède à une tâche de H et simultanément en précède une autre appartient également à H.

Le principe des méthodes d'ALB consiste à générer les postes d'assemblage lors d'une partition des nœuds (tâches d'assemblage) d'un graphe de précédence. Un sous-ensemble P de tâches peut être affecté à un poste d'assemblage s'il existe deux sous-ensembles ayant la structure de poste d'assemblage, tels que P peut s'obtenir comme leur différence. P est appelé *poste candidat*.

Exemple I-1 :

Étant donné le graphe de précédence de la figure I-3, nous illustrons sur la même figure la répartition en postes candidats.

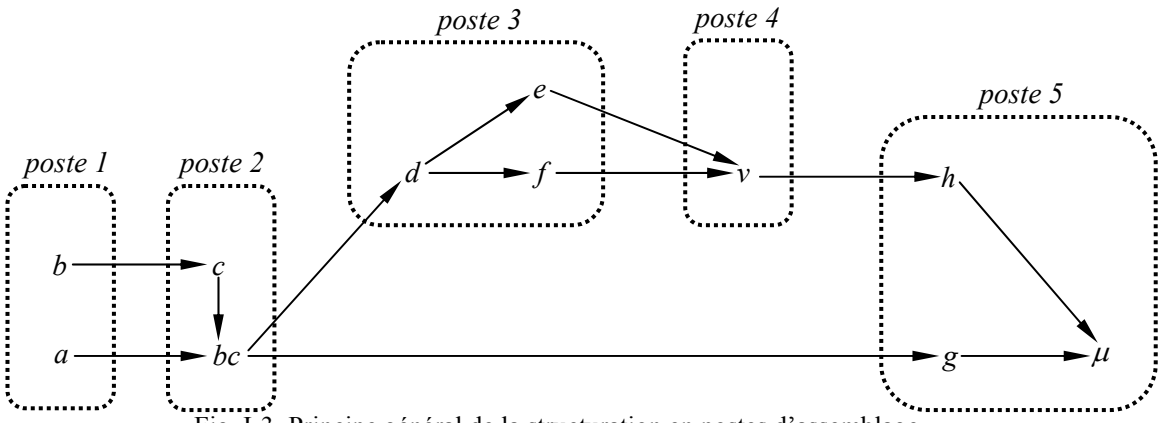


Fig. I-3. Principe général de la structuration en postes d'assemblage

Nous obtenons d'abord la délimitation de quelques structures de postes :

$H_1 = \{a, b\}$ $H_2 = \{a, b, c, bc\}$ $H_3 = \{a, b, c, bc, d, e, f\}$

$H_4 = \{a, b, c, bc, d, e, f, v\}$ $H_5 = \{a, b, c, bc, d, e, f, v, h, g, \mu\}$

Il en résulte un découpage en cinq postes candidats :

$P_1 = H_1 = \{a, b\}$ $P_2 = H_2 - H_1 = \{c, bc\}$

$P_3 = H_3 - H_2 = \{d, e, f\}$ $P_4 = H_4 - H_3 = \{v\}$ $P_5 = H_5 - H_4 = \{h, g, \mu\}$

Pour des raisons de simplicité, on évite parfois la formulation de la contrainte topologique sous forme de poste candidat, en préférant sa traduction par deux contraintes : l'affectation de chaque tâche à un seul poste d'assemblage et l'affectation des tâches qui se précèdent respectivement aux postes qui se précèdent.

Les méthodes de conception d'ALB utilisent deux grandes classes d'algorithmes : les ***algorithmes exacts*** et les ***algorithmes heuristiques***. Lorsque le problème d'ALB est avant tout un problème d'optimisation combinatoire NP-complet, les algorithmes exacts présentent l'inconvénient d'une trop grande complexité. Une solution alternative est offerte par le développement des algorithmes heuristiques.

En 1955 Salveson a proposé la programmation linéaire comme premier algorithme exact appliqué à l'équilibrage des lignes d'assemblage. Cette technique conduisait à des solutions infaisables dans la pratique. Depuis lors, des nouvelles techniques ont été utilisées dans les ***algorithmes exacts*** :

- la programmation entière (*integer programming*) ;
- la programmation dynamique (*dynamic programming*) ;
- les méthodes de buts programmés (*goal programming*) ;
- la technique division-élagage (*branch-and-bound*).

Nous allons passer en revue les caractéristiques de quelques méthodes typiques.

La *programmation entière* a été premièrement adoptée dans une méthode d'ALB par Bowman (1960). Elle apportait comme nouveauté l'introduction d'une variable binaire de décision et d'un coût de pénalité. Dans l'approche de Talbot et Patterson (1984) cette variable n'est plus définie comme binaire, mais comme le numéro du poste d'assemblage x_i auquel la tâche i est affectée. Un principe analogue est employé dans [GUNT 83], où on trouve la formulation des objectifs et des contraintes sous forme de buts ordonnés par importance et la minimisation de toute déviation par rapport à ces buts.

Dans [PINT 83] on propose l'utilisation de la *programmation entière en tenant compte du coût économique*, basée sur l'assertion qu'il existe, pour chaque tâche, un équipement de base de moindre coût offrant une performance satisfaisante. Chaque autre solution pour équiper le système d'assemblage est supérieure comme performance, mais également caractérisée par un coût d'investissement plus élevé. L'objectif de l'algorithme est de trouver la solution alternative qui minimise la différence de coût d'investissement par rapport à la solution de base. Cela est un problème d'optimisation soumis aux contraintes suivantes :

- respect du temps de cycle effectif (le plus grand temps de travail des postes d'assemblage) ;
- l'affectation de chaque tâche à un seul poste de travail ;
- respect des contraintes d'antériorité entre tâches.

Jackson (1956) a premièrement proposé la *programmation dynamique* afin de résoudre le problème d'ALB. L'objectif de l'algorithme est de trouver le nombre minimal de postes pour l'affectation des tâches d'un ensemble donné I sans violer la contrainte temporelle.

Ce nombre est noté par $D(I)$, ou également par n^*, lorsqu'on fixe I. On définit les sous-ensembles k-maximaux d'un ensemble donné de tâches : $H \subseteq I$ est dit k-maximal si $D(H)=k$ et s'il n'existe pas un autre sous-ensemble H' tel que $H \subseteq H'$ et $D(H')=k$. La procédure de recherche trouve les sous-ensembles n^*-maximaux d'une manière récurrente, en élaborant à chaque pas les sous-ensembles $(k+1)$-maximaux par regroupements des sous-ensembles k-maximaux et des sous-ensembles 1-maximaux de la partie restante du graphe de précédence.

Toute méthode de résolution du problème d'ALB est essentiellement une recherche sur l'arbre de toutes les solutions possibles. La manière de faire cette recherche fournit la différence entre les algorithmes exacts et les algorithmes heuristiques. Face à l'exploration exhaustive proposée par les premiers, les ***algorithmes heuristiques*** cherchent à ne développer qu'une partie des nœuds de l'arbre, en espérant pouvoir trouver une *solution sous-optimale*.

Dans le problème d'ALB, le choix des nœuds à explorer est appelé **règle d'affectation**, lorsqu'il s'agit à chaque pas du choix de la tâche suivante qui doit être affectée au poste en train de constitution. Deux contraintes influencent ce choix :

- le respect du temps de cycle imposé ;
- la nécessité d'avoir antérieurement affecté toutes les tâches qui précèdent celle-ci.

Nous illustrons dans l'exemple suivant le principe de la règle d'affectation au cours d'une recherche heuristique.

Exemple I-2 :

Reprenons le graphe de précédence de la figure I-3. Dans la figure I-4 nous avons supposé que les postes *1* et *2* sont déjà constitués et que la tâche *d* soit déjà affectée au poste *3*. Supposons également que les temps d'exécution des tâches *d*, *e*, *f* et *g* soient :

$t_d = 2$, $t_e = 4$, $t_f = 5$, $t_g = 4.5$

et que le temps de cycle imposé soit $T_c = 10$.

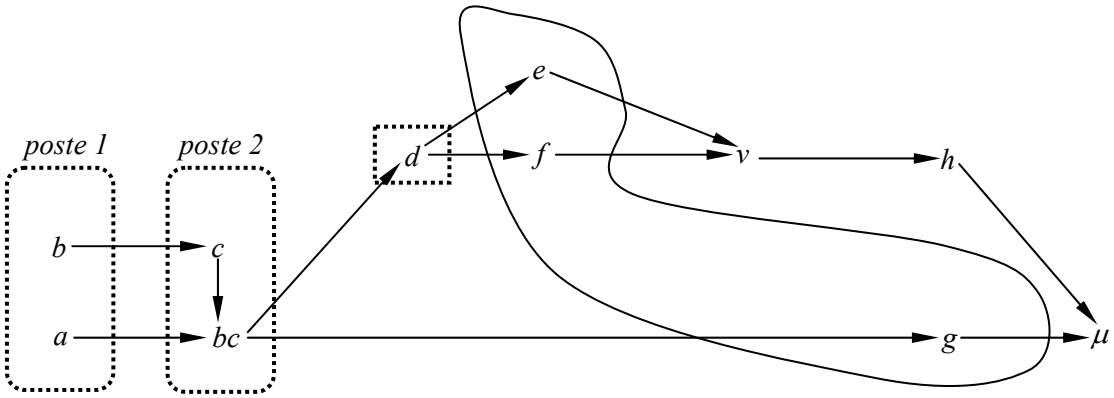


Fig. I-4. Principe de la recherche heuristique : la règle d'affectation des tâches

Pratiquement, c'est la deuxième contrainte qui agit premièrement sur la constitution de la liste des tâches candidates : nous allons les choisir dans l'ensemble des successeurs des tâches déjà affectées. Il en résulte la liste $\{e, f, g\}$, qui est soumise ensuite à la première contrainte. Nous constatons que :

$$t_d + t_e = 6 \le T_c \qquad t_d + t_f = 7 \le T_c \qquad t_d + t_g = 6.5 \le T_c$$

et, donc, toute tâche de la liste ci-dessus, si l'on affecte au poste *3*, ne conduit pas au dépassement du temps de cycle imposé.

Dans un premier temps, on peut considérer que l'affectation se fait *aléatoirement* [ARCU 66], selon le principe du choix d'une tâche quelconque dans la liste des tâches candidates. Évidemment, dans l'absence de la garantie de l'optimalité de la solution, il y a peu de chances qu'une seule recherche conduise à une bonne solution.

Étant lui-même conscient de l'inefficacité de son algorithme, A. Arcus a proposé un *algorithme à la règle prioritaire d'affectation de tâches* [ARCU 66], basée sur un ensemble de neuf règles à établir un ordre de priorités sur l'ensemble des tâches. Certaines règles sont proches ou même coïncident avec les règles proposées par d'autres auteurs. Par exemple, accorder la priorité à la tâche qui a le plus grand temps d'exécution au sein de la liste candidate [MOOD 65], ou à celle pour laquelle la somme de son temps d'exécution et de ceux des tâches qui la suivent est la plus grande [HELG 61], ou bien aux tâches dont le nombre des prédécesseurs est le plus petit [KILB 61], etc.

Comme le problème d'ALB regarde essentiellement l'optimisation du découpage en postes, les nouveaux travaux de recherche dans ce domaine peuvent être caractérisés par l'utilisation des nouvelles méthodes d'optimisation, surtout stochastiques. Nous reprenons en détail ce sujet au cours du chapitre IV de ce travail (voir IV.3.3).

I.2.2.2 Méthodes C.S.D.L.

Les travaux de recherche menés au sein du Charles Stark Draper Laboratory proposent des méthodes de conception des systèmes d'assemblage à moindre coût économique à partir des *séquences d'assemblage*. Ces travaux ont été le point de départ des concepts développés au Laboratoire d'Automatique de Besançon (L.A.B.) [PERR 92]. Il est à noter que les méthodes C.S.D.L. sont facilement adaptables au cas multiproduit, lorsqu'elles ne distinguent généralement pas les tâches d'assemblage permettant la réalisation de différents produits.

La formulation des critères d'optimisation économique est basée sur l'analyse du coût d'un système d'assemblage, qui est composé de quatre parties [GUST 88] :

- le *coût fixe*, qui est l'investissement des équipements principaux dépensé au moment de l'installation et amorti pendant une période déterminée ;
- le *coût variable*, qui provient principalement des travaux effectués par les opérateurs humains ;
- le *coût matériel*, qui provient de la consommation des matériels non amortis ;
- le *coût de gestion*, qui regroupe les coûts d'administration et de gestion de l'entreprise.

Le coût matériel et le coût de gestion peuvent être ramenés à des coûts fixes. Donc, le coût de tout opérateur est formé d'un coût fixe et d'un coût variable, le dernier étant proportionnel au temps d'exploitation.

Il existe deux types d'algorithmes associés aux méthodes C.S.D.L. : la ***programmation entière*** [GRAV 79][GRAV 83][GRAV 88] et la ***recherche systématique*** [WHIT 86][HOLM 87][GRAV 88].

En ce qui concerne la *programmation entière*, une première méthode systématique a été proposée en 1979 par S.C. Graves et D.E. Whitney [GRAV 79]. Étant dédiée au cas monoproduit, cette méthode permet une affectation des tâches aux équipements, qui est optimale du point de vue des coûts fixes et des coûts variables.

Cette approche a été enrichie en 1983 par Graves et B.V. Lamar [GRAV 83], en tenant compte des temps de changement d'outils et en permettant des passages multiples d'un produit intermédiaire dans un même poste.

La formalisation générale donnée par S.C. Graves et C.A. Holmes [GRAV 88] regarde le cas multiproduit, étant soumise aux contraintes du cadre général de conception (voir I.2.1) et ayant comme objectif la minimisation du coût économique.

La *recherche systématique* est employée dans les travaux de D.E. Whitney, C.A. Holmes et S.C. Graves. Ces auteurs ont proposé une méthode d'exploration systématique de toutes les solutions de structuration d'une ligne d'assemblage, qui est applicable aussi bien dans le cas monoproduit, que dans le cas multiproduit.

Dans ce dernier cas, il s'agit d'une *famille iso-structurelle de produits*, dont la définition repose sur l'existence de groupes de composants élémentaires jouant le même rôle dans la structure de leurs produits [DUFR 91]. On parle dans ce cas de *groupes fonctionnels de composants*. Si Q est l'ensemble des composants élémentaires et des groupes fonctionnels de composants, alors il existe une relation binaire sur $Q \times Q$, qui détermine une partition de Q en classes d'équivalence.

Le modèle des processus d'assemblage utilisé dans la recherche systématique est une *séquence d'assemblage* pour un produit, ou la réunion des séquences d'assemblage associées aux produits d'une même famille. Dans le cas monoproduit, la séquence d'assemblage est, en fait, un graphe de précédence linéaire, où toutes les tâches sont strictement ordonnées dans le temps. On utilise également le terme de "gamme" pour désigner une séquence.

Les autres données d'entrée de la méthode sont :

- le temps de cycle imposé, correspondant à une productivité imposée ;
- les opérateurs qui peuvent exécuter les tâches d'assemblage ;
- pour chaque tâche, les couples opérateur-outil pouvant l'effectuer ;
- pour chaque couple opérateur-outil, le temps d'exécution d'une tâche ;
- le coût fixe annuel de chaque opérateur ;
- le coût variable/heure de chaque opérateur ;
- le coût annuel d'un outil, dépendant sur la tâche exécutée et sur l'opérateur affecté à cette tâche ;
- le temps de changement d'outils comme fonction d'opérateur.

La recherche se fait en deux phases :

1) énumérer et représenter graphiquement les séquences d'assemblage pour en sélectionner une, considérée comme optimale ;
2) trouver le système d'assemblage de coût minimum – défini par l'*affectation d'équipements* et le *découpage en postes de travail* – qui met en œuvre une des séquences résultées dans la première phase.

Les deux problèmes de la deuxième phase – l'affectation et le découpage – sont résolus simultanément à l'aide d'une méthode d'optimum qui énumère et évalue implicitement tous les postes candidats. La méthode de génération est la même que dans les méthodes d'ALB : chaque poste candidat est une différence entre deux structures de poste d'assemblage (voir I.2.2.1). Nous donnons un exemple suggestif pour le cas multiproduit.

Exemple I-3 : [GRAV 88]

Considérons le cas d'une famille de deux produits :

- le produit *A* nécessite les tâches *1*, *2*, *3*, *5*, *6*, *8*, *9*, *10*, *12* ;
- le produit *B* nécessite les tâches *1*, *2*, *4*, *5*, *6*, *7*, *10*, *11*, *12*.

Le modèle des processus d'assemblage est obtenu comme réunion des séquences d'assemblage destinées à la réalisation des produits *A* et *B* (voir figure I-5). Nous remarquons que cette réunion ressemble à un graphe de précédence.

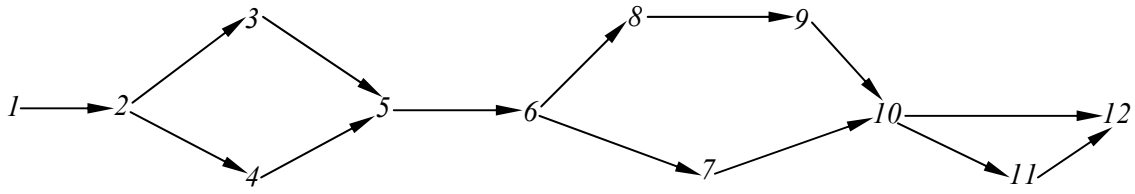


Fig. I-5. Réunion de deux séquences d'assemblage pour deux produits

Nous illustrons le principe de la détermination des structures d'assemblage sur le modèle de la figure I-5. Par exemple, la différence entre deux telles structures, *L* et *G*, donne le poste candidat {*6*, *8*, *9*} (voir figure I-6).

L'ensemble des postes candidats ainsi générés peut être réduit en ne considérant que les postes candidats faisables. On considère qu'un poste candidat est *faisable* si et seulement si l'ensemble de tâches affectées à ce poste peut être effectué par au moins un opérateur pendant le temps de cycle imposé. On trouve ici l'expression de la contrainte temporelle : le temps de cycle de tout poste doit être inférieur au temps de cycle imposé au système. Le calcul du temps de cycle d'un poste peut prendre en compte les changements éventuels d'outils entre tâches.

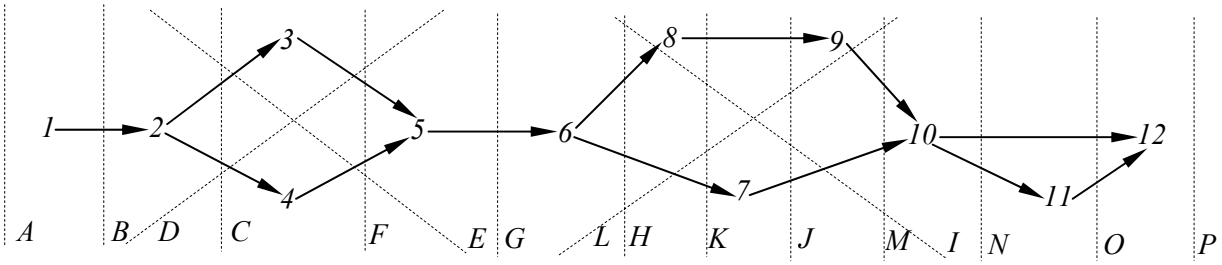


Fig. I-6. Obtention des structures des postes donnant des postes candidats

La deuxième phase de la recherche systématique peut être ensuite détaillée en trois grandes étapes :

a) énumérer tous les postes de travail potentiels ;
b) trouver, pour chaque poste, l'affectation de ressources qui assure le coût minimum ;
c) trouver la ligne d'assemblage de coût minimum (chercher la partition de l'ensemble de tâches d'assemblage de coût minimum).

On détaille l'étape c) à partir du fait que pour chaque poste candidat faisable on peut sélectionner un opérateur de moindre coût. Ainsi, il est possible de constituer un *graphe de recherche* où :

- chaque nœud est une structure de poste d'assemblage ;
- chaque arc entre deux nœuds représente un poste candidat faisable.

Puisqu'à chaque arc est associé un coût économique, le problème de la recherche du système de moindre coût est réduit au problème classique du plus court chemin, résoluble par l'algorithme de Dijkstra.

L'algorithme de C.A. Holmes a été étendu au cas de la collaboration de plusieurs équipements dans un même poste [WHIT 88], où on propose comme résultat non plus un, mais plusieurs systèmes de coût minimal.

Un désavantage de la structuration d'un système d'assemblage selon le critère du coût minimum est l'obtention potentielle de solutions difficiles à implanter du point de vue spatial.

I.2.3 Autres méthodes

Sous ce titre nous présentons les grandes axes de recherche dans le domaine de la conception des systèmes d'assemblage.

- *Génération et sélection des processus d'assemblage*

À présent nous pouvons dire que les algorithmes de génération automatique des processus d'assemblage sont parfaitement mis au point – le travail [ALMG 89] en est un état de l'art – mais cela n'est pas le cas au niveau de la sélection des processus d'assemblage.

L'obtention des processus d'assemblage commence avec le *modèle du produit*. Dans [DELC 89b] on trouve un état de l'art sur la modélisation du produit. Cela dépend de trois éléments : les outils utilisés, les informations fournies et l'objectif envisagé. Dans la littérature il en existe deux approches [DELC 90a] :

- *modélisation géométrique*, soutenue par les logiciels de CAO ;
- *modélisation relationnelle*, considérée comme incomplète.

Actuellement, la plus connue et la plus utilisée des approches de modélisation est la *modélisation du produit fini à l'aide des caractères* – premièrement proposée dans la thèse [BOUJ 84], reprise dans [HOME 88, 89, 90, 91], [FAZI 88], [HENR 89] – qui est une modélisation relationnelle. La définition du modèle qui en résulte est donnée dans I.2.4.2.

Les travaux [JENT 83], [JEAN 86], [FIGO 88] proposent la génération des processus d'assemblage basée sur le démontage du produit. Dans [ELMA 89], à partir d'une représentation du produit similaire à son modèle fonctionnel, on obtient les processus d'assemblage par des règles expertes. La méthode de génération développée dans [BOUJ 84], qui est reprise dans [LUI 88], est dédiée au cas monoproduit, tandis que le cas multiproduit est traité dans [CAMP 89].

Différents types de modèles sont utilisés lors de la génération des processus d'assemblage :

- graphes de précédence [CHOW 90][GREW 90],
- arbres d'assemblage [HENR 89],
- graphes ET/OU (qui sont des représentations collectives des arbres d'assemblage) [HOME 88,89,90,91][LEVI 88],
- *P-Q-R* arbres [BOOT 76][BAPT 91],
- réseaux de Petri non autonomes, temporisés ou non [DAVI 89],
- réseaux de Petri colorés [ALLA 87][GENT 88a, 88b],
- multigraphes bivalués [CARL 88],
- équations récursives dans un dioïde [COHE 85, 90, 91], etc.

Dans [CHEN 96] on trouve une présentation comparative de quatre premiers types de modèles listés ci-dessus, les plus souvent utilisés dans la conception. Le même travail présente également une évaluation générale de ces modèles du point de vue de l'aisance d'exploitation dans la conception, d'où on conclut que les modèles les plus adaptés aux buts de conception sont les *arbres d'assemblage* et les *graphes de précédence*.

- *Choix des équipements et affectation des tâches aux équipements*

Les travaux récents proposent des approches basées sur la programmation dynamique [GUST 88] ou sur des règles heuristiques et implémentées par des systèmes expert [WINT 85][SIMS 88][KUSI 90].

- *Évaluation des performances temporelles des systèmes et modèles de simulation*

Les travaux effectués au Laboratoire d'Automatique de Grenoble ([L.A.G. 90], [L.A.G. 91]) et à l'Institut du Nord ([GENT 88a], [GENT 88b]) s'inscrivent dans cette axe de recherche.

Nous pouvons remarquer aussi d'autres directions de recherche, comme, par exemple, l'évaluation financière des systèmes, la construction d'équipements, l'implantation des équipements d'un système d'assemblage, etc.

I.2.4 Méthode L.A.B.

I.2.4.1 Stratégie de la méthode

La stratégie de la méthode de conception développée au Laboratoire d'Automatique de Besançon (L.A.B.) est de générer ***tous*** les systèmes d'assemblage qui correspondent aux données de départ, puisque la recherche d'une solution "optimale" est considérée comme illusoire. La variante la plus convenable est finalement choisie par l'expert humain. On élimine au fur et à mesure les systèmes inacceptables, lors de l'évaluation de leurs performances. Ces évaluations sont faites dans toutes les étapes de la méthode L.A.B.

Cette stratégie induit un phénomène impossible à contourner : l'explosion combinatoire. Elle peut être cependant plus ou moins maîtrisée par des contraintes intervenant au cours du déroulement de la conception.

I.2.4.2 Principe général. Étapes principales

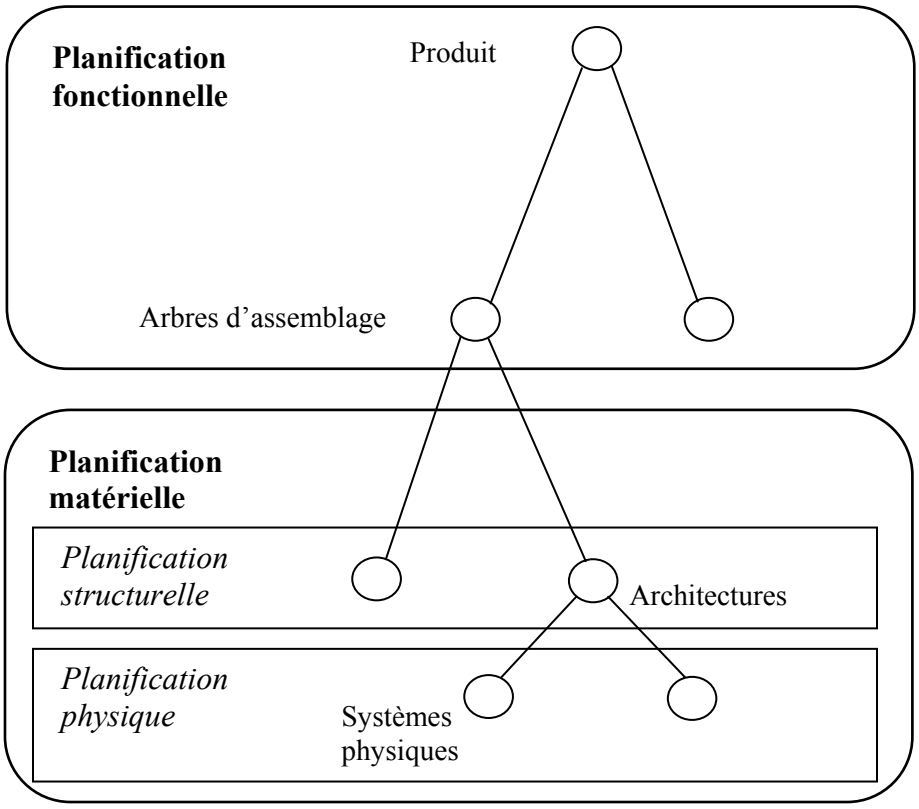


Fig. I-7. Méthode L.A.B. de conception des systèmes d'assemblage

Le principe général de la méthode est basé sur la décomposition du problème de conception en *deux grandes parties* [MÎNZ 95] (voir figure I-7) :

- **planification fonctionnelle**, qui donne comme résultat l'ensemble de tâches qui doivent être effectuées, ainsi que leur organisation temporelle ;
- **planification matérielle**, dont le résultat est un ensemble d'équipements destinés à réaliser les tâches précédemment définies, en respectant l'objectif de production et à un coût aussi réduit que possible ; en pratique, cette étape est divisée en deux phases :
  - *planification structurelle*, lors de laquelle on définit les types d'équipements et l'architecture globale des systèmes candidats ;
  - *planification physique*, conduisant à la définition complète des systèmes retenus.

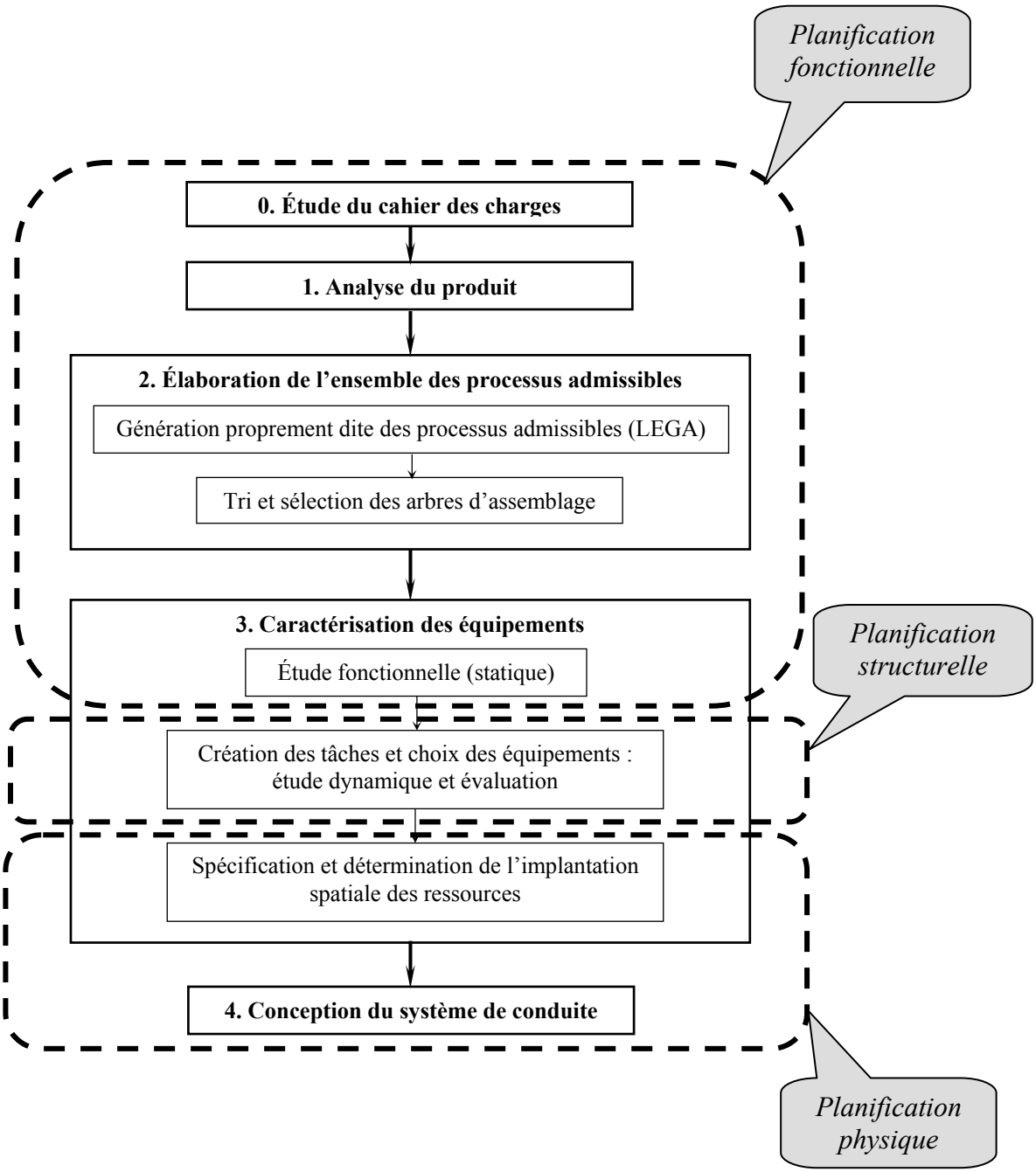


Fig. I-8. Étapes principales de la méthode L.A.B.

Ensuite, la décomposition du problème de conception conduit à la résolution des sous-problèmes à travers *cinq étapes principales*, conformément à l'organigramme de la figure I-8. Dans la même figure nous avons montré également comment chacun des sous-problèmes s'attache aux grandes étapes de la méthode L.A.B., présentées à la figure I-7.

Chacune des étapes ci-dessus amène le concepteur à remettre en cause la conception, lors de certains choix faits en fonction des stratégies d'assemblage adoptées. Ces processus de décision n'influencent pas nécessairement sur le futur système d'assemblage, mais ils peuvent influencer sur la modélisation du produit. Il faut, donc, recommencer certaines étapes précédentes de la méthode avec un autre modèle du produit, c'est à dire avec un produit modifié. Nous présentons ensuite l'essentiel de chacune de ces étapes.

L'***étude du cahier des charges (étape 0)*** met en évidence les premières contraintes à respecter dans la conception d'un système d'assemblage. Il s'agit, par exemple, des caractéristiques de l'atelier, des matériels existants, des limitations logistiques, du degré d'automatisation, etc.

L'***analyse du produit (étape 1)*** a pour objectif sa modélisation. Le modèle du produit est nécessaire à la modélisation des processus d'assemblage, conformément à l'ordre logique :

analyse du produit → *modèle du produit* → *modèle des processus*

Deux types de modèles du produit ont été successivement utilisés dans la méthode L.A.B. :

- le **modèle fonctionnel** [BOUJ 84]
- et le **modèle opératoire** [HENR 89].

Ces deux modèles ne sont pas contradictoires, mais complémentaires.

Modèle fonctionnel

L'existence d'une **liaison fonctionnelle** entre deux composants élémentaires d'un produit donné est en correspondance biunivoque avec l'existence d'au moins une liaison mécanique entre ces deux composants.

Définition I-1 : graphe des liaisons fonctionnelles

Le graphe des liaisons fonctionnelles du produit P est un graphe simple et non orienté $G = (C, L)$, où :

- l'ensemble des nœuds du graphe est l'ensemble C des composants élémentaires du produit P ;
- l'ensemble des arêtes du graphe est l'ensemble L des liaisons fonctionnelles du produit P.

Le graphe des liaisons fonctionnelles constitue le modèle fonctionnel du produit.

Modèle opératoire

Ce modèle est basé sur un traitement différencié sur l'ensemble des liaisons fonctionnelles et sur la mise en évidence de l'ensemble des caractères qui définissent le produit. Ils peuvent être de nature géométrique, physique ou complémentaire.

S'il existe un contact entre deux composants élémentaires c_i et c_j dans au moins l'un des états du produit fini, alors nous disons qu'il y existe une **liaison géométrique**. L'existence d'une liaison géométrique et d'une énergie de cohésion, indépendante de la pesanteur entre c_i et c_j, caractérise la présence d'une **liaison physique**. Une solidarisation est un procédé qui permet d'établir une ou plusieurs liaisons physiques.

Les **caractères complémentaires** représentent tous les caractères des produits, autres que les liaisons.

Les **caractères non géométriques** sont évidemment les caractères physiques et les caractères complémentaires. On définit une application *h* qui fait correspondre à chaque caractère non géométrique l'ensemble des composants sur lesquels il porte.

En analogie avec le graphe des liaisons fonctionnelles, à tout produit P on associe le ***graphe*** non orienté ***des liaisons géométriques***, $G'= (C, L')$, dont l'ensemble des sommets correspondent de façon biunivoque aux composants élémentaires de P et l'ensemble de ses arêtes aux liaisons géométriques de P ($L' \subseteq L$).

Définition I-2 : ***modèle opératoire***

Tout produit fini est décrit par un 5-uplet $<C, L', \Sigma, \Delta, h>$, où :

- C et L' correspondent au graphe des liaisons ;
- Σ est l'ensemble des solidarisations ;
- Δ est l'ensemble des caractères complémentaires ;
- h définit des conditions nécessaires d'établissement de tout caractère non géométrique.

Le modèle opératoire est appelé aussi ***modèle du produit fini à l'aide des caractères***. Dans l'étape d'analyse du produit on collecte les informations permettant de déterminer les éléments du modèle opératoire.

L'***élaboration des processus d'assemblage admissibles (étape 2)*** commence avec la génération proprement dite des processus admissibles sous forme d'*arbres d'assemblage*. Dans ce but on utilise le logiciel LEGA (Logiciel d'Élaboration des Gammes d'Assemblage), conçu par J.M. Henrioud [HENR 89], qui reçoit comme donnée d'entrée le modèle opératoire du produit. Dans II.2.2 nous allons détailler les étapes de ce passage.

La définition de l'arbre d'assemblage repose sur une définition plus formelle de l'opération d'assemblage.

Une **opération d'assemblage** est généralement représentée par une paire non ordonnée$\{C, e\}$, où :

- C est un constituant ;
- e est soit un constituant, soit un caractère non géométrique.

Un constituant est soit un composant élémentaire, soit un sous-assemblage caractérisé par un ensemble composé à la fois de composants élémentaires et de caractères non géométriques.

Un a**rbre d'assemblage** est une arborescence dont :

- la racine est le produit fini,
- les nœuds non terminaux sont des sous-assemblages,
- les feuilles sont des composants élémentaires et des caractères non géométriques,

et telle que tout sous-assemblage situé à un nœud non terminal est réalisé par une opération à partir de ses deux successeurs.

Tout processus admissible ne contient que de tâches *faisables*, c'est à dire de tâches qui satisfont aux contraintes d'assemblage (géométriques, matérielles, de stabilité).

Dans [MÎNZ 95] on propose la transformation d'un arbre d'assemblage en un modèle plus détaillé du processus d'assemblage, le *graphe d'assemblage*, qui opère avec des opérations et des constituants localisés (voir aussi II.2.3). Nous verrons dans le chapitre II que le graphe d'assemblage est un graphe de précédence particulier.

L'ensemble des arbres générés dans la phase antérieure est soumis à une sélection selon des critères issus de différentes stratégies d'assemblage, qui peuvent être relatifs aux : sous-assemblages, composants, liaisons entre composants, etc. (voir II.2.2).

L'étude statique de la ***caractérisation des équipements (étape 3)*** doit déterminer l'ensemble des opérations positionnelles (transferts ou réorientations des sous-assemblages) et associer à chaque opération un type d'équipement. Dans [OLIV 86] on propose la représentation du processus par des *schémas fonctionnels*.

Un schéma fonctionnel résulte d'un arbre d'assemblage auquel on ajoute les opérations positionnelles et dont la signification change : les nœuds représentent les opérations (qui maintenant sont des opérations localisées) et les arcs représentent les constituants (localisés aussi).

La symbolique utilisée est illustrée à la figure I-9. Les trois cas correspondent à une :

a) opération *géométrique* d'ordre 2, où il s'agit de l'établissement d'un ensemble *L'* de liaisons géométriques entre deux constituants : C_1 (qui reste fixe, étant appelé *primaire*) et C_2 (qui est mobile, étant appelé *secondaire*), qui conduit au constituant résultant C_3 ; un constituant primaire qui est un composant élémentaire s'appelle *composant de base* ;

b) opération *non géométrique*, où le constituant C_1 se transforme en C_2, en acquérant un ensemble de caractères non-géométriques, $\Sigma' \cup \Delta'$, par la mise en œuvre d'un procédé *P* ;

c) opération *positionnelle*, où *C* est le constituant soumis à une réorientation.

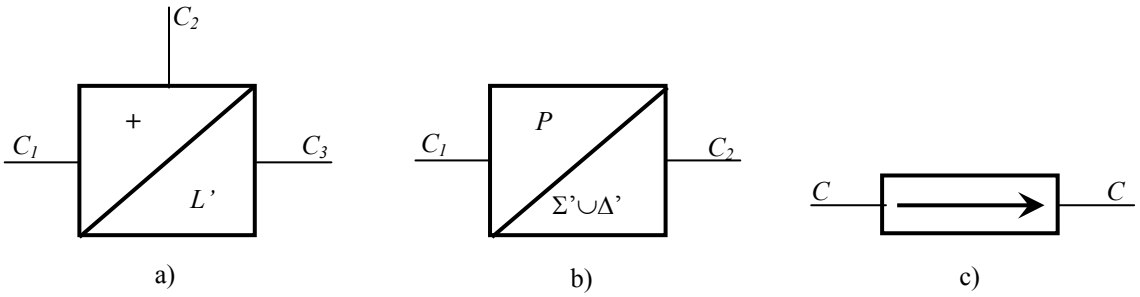


Fig. I-9. Symbolique des schémas fonctionnels

À noter qu'un schéma fonctionnel décrit les fonctions du système d'assemblage sans prendre en compte ni les équipements du système, ni les temps opératoires.

Le logiciel AISE (Aides à l'Implantation et à la Spécification des Équipements), élaboré au L.A.B., comporte un outil d'interface permettant la génération, la représentation et la manipulation des schémas fonctionnels.

La deuxième phase de l'***étape 3*** a comme but l'*étude dynamique* (temporelle) du système, à l'aide d'une simulation plus ou moins précise. Deux éléments doivent résulter de cette phase : la spécification des équipements qui exécutent les opérations du schéma fonctionnel et le découpage en postes de tout le système d'assemblage.

Ce dernier problème peut être résolu par l'intermédiaire de DPSA (Découpage en Postes d'un Système d'Assemblage), logiciel conçu au L.A.B., qui est dédié au cas monoproduit [MÎNZ 95]. L'approche multiproduit constitue une généralisation de l'approche monoproduit [MÎNZ 95] (voir aussi IV.2.2).

Le logiciel AISE comporte un module permettant la simulation des systèmes préfigurés par les choix faits. Il est basé sur une représentation du schéma fonctionnel et de l'affectation opérations-équipements par un réseau de Petri temporisé [CHAP 88].

La proposition d'une implantation spatiale du système d'assemblage est le but de la dernière phase de l'***étape 3***. Préalablement, il est possible de décider de ce qui peut être automatisé et comment au sein de chaque système obtenu dans l'étape précédente.

On procède ensuite à une simulation fine des solutions retenues, au moyen d'outils appropriés. Les travaux [BARR 91a], [TEYS 91] et [BARR 91b] ont été menés au L.A.B. dans ce sens.

À l'égard de l'étape 3 nous devons mentionner également la thèse [BONN 94], où l'auteur présente une base de données destinée à la caractérisation des équipements. Une telle base constitue un outil indispensable à la conception d'un système d'assemblage.

Dans la dernière étape de la méthode L.A.B. il s'agit de la ***conception du système de conduite (pilotage) (étape 4)***, qui englobe des algorithmes de conduite correspondant aux différents niveaux de commande. Parmi eux, citons :

- la commande au niveau des équipements, signifiant le *contrôle en temps réel* des équipements issus de l'étape antérieure ;
- la commande au niveau de la stratégie de *pilotage* de l'atelier tout entier ;
- la *gestion de production* de l'atelier en fonction des commandes passées par la clientèle.

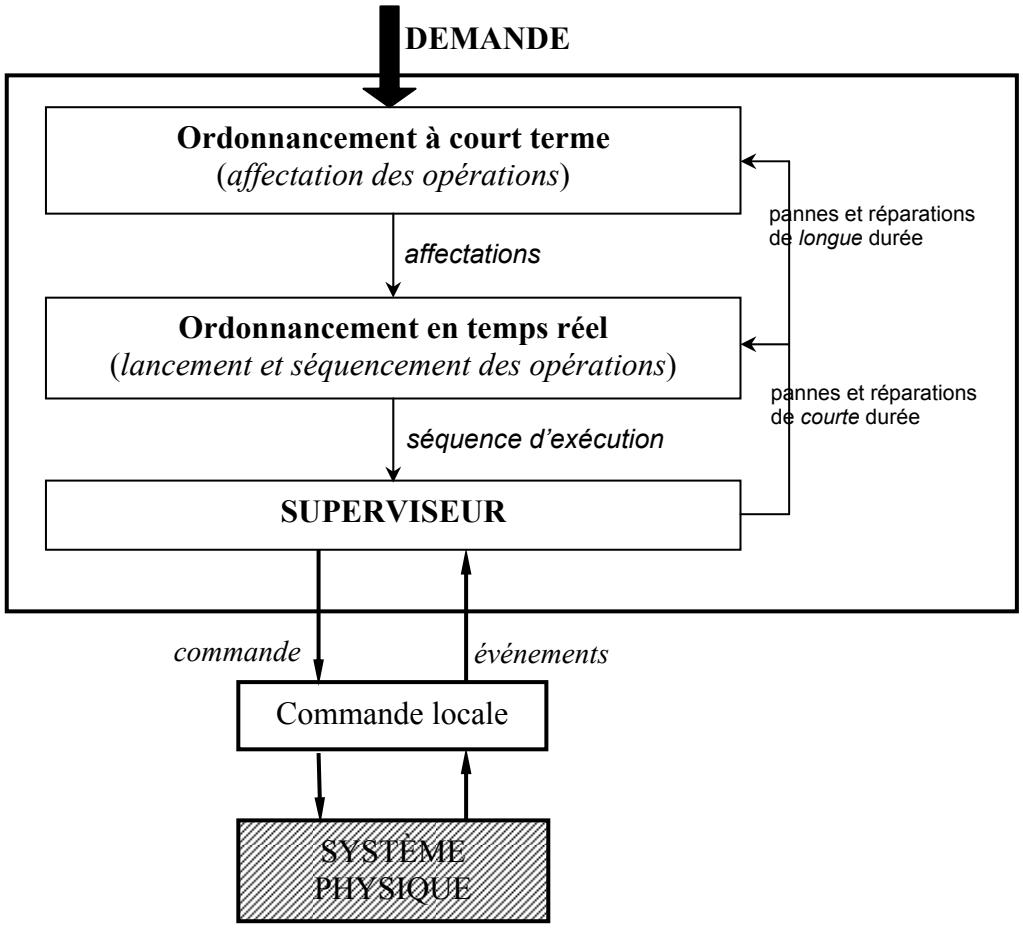


Fig. I-10. Structure d'un système de pilotage

Le terme "pilotage" regarde les systèmes de production en général. Dans la thèse [YIN 95], élaborée au L.A.B., on trouve une approche hiérarchisée du *pilotage réactif* des systèmes d'assemblage flexibles soumis aux pannes de ressources, basée sur la méthode de l'*équilibrage dynamique*. Les termes "réactif" et "dynamique" désignent la capacité des

algorithmes de conduite élaborés de fonctionner *en temps réel*, de sorte que le système d'assemblage ait une réponse bonne et rapide (une bonne réactivité) aux perturbations externes (la demande) ou internes (dysfonctionnements des ressources).

Dans la figure I-10 est présentée la structure d'un système de pilotage. Nous avons mis en évidence les éléments du pilotage : l'ordonnancement à court et à très court terme, ainsi que la commande du système de production.

Quelques algorithmes présentés dans la thèse [MÎNZ 95] touchent au problème de la conduite des systèmes d'assemblage. Plus précisément, ces algorithmes regardent le problème de la *restructuration* d'un système d'assemblage en cas de panne d'un porteur (réaffectation réactive des tâches), étant, donc, utilisables pour le pilotage.

I.2.4.3 Algorithmes

Quant aux algorithmes utilisés dans la méthode L.A.B. de conception des systèmes d'assemblage, nous mentionnons deux approches utilisant deux modèles différents des processus d'assemblage : l'approche basée sur les *arbres d'assemblage linéaires* et l'approche basée sur les *graphes d'assemblage*.

La première approche utilise en tant que processus d'assemblage un ***arbre d'assemblage linéaire***, pour lequel le produit fini est obtenu par l'ajout successif des composants élémentaires sur une même base (voir II.2.2).

La conception se déroule à travers deux étapes :

- génération des postes candidats par la partition successive des opérations composant l'arbre d'assemblage linéaire ; un poste candidat est défini par deux conditions à satisfaire :
  - il n'existe qu'au moins un opérateur pouvant effectuer l'ensemble de tâches affectées ;
  - le temps de travail de ce poste pour chacun des opérateurs valables ne dépasse pas le temps de cycle imposé ;
- recherche du système de moindre coût par la technique du chemin le plus court.

À chaque poste candidat on associe un coût économique, qui est celui de l'opérateur le moins cher. On construit un graphe de recherche sous forme d'une "pyramide de postes" dans lequel la recherche du chemin le plus court nous conduit vers le système de moindre coût. Un exemple se trouve dans [PERR 92].

L'approche basée sur les ***graphes d'assemblage*** [MÎNZ 95] repose sur un nouveau modèle des processus d'assemblage, qui réunit les avantages des graphes de précédence et des arbres d'assemblage, en formulant une expression originale de la contrainte topologique (c'est à dire la définition et la généralisation des postes candidats). Cette expression s'appuie sur les notions de "branche" et de "segment".

Dans II.2.3 nous reprenons le modèle de type graphe d'assemblage, en soulignant sa liaison d'une part avec les arbres d'assemblage, et d'autre part avec les graphes de précédence. Un exemple d'obtention des graphes d'assemblage à partir des arbres d'assemblage sera également présenté.

Nous nous limitons pour l'instant à montrer l'utilité du graphe d'assemblage dans la démarche de conception des systèmes d'assemblage.

Une *branche* au sein d'un graphe d'assemblage est l'ensemble de nœuds visités par un chemin qui joint une feuille au premier nœud rencontré qui est le prédécesseur gauche d'un nœud, ou à la racine. La figure I-11 montre un graphe d'assemblage composé de deux branches : $B_1 = <a, b>$ et $B_2 = <c, \mu>$.

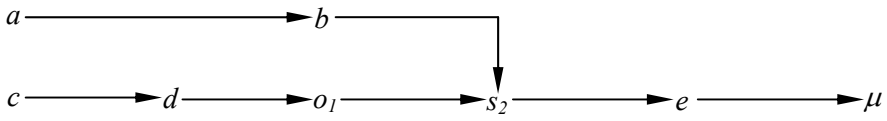


Fig. I-11. Exemple de graphe d'assemblage

Un *segment de branche* est un ensemble formé par un ou plusieurs nœuds qui ont des positions consécutives sur une branche. Dans le cas du graphe d'assemblage de la figure I-11, $<o_1, e>$ est un segment de la branche B_2.

Compte tenu de ces définitions, l'auteur a démontré que tout poste candidat est une réunion de segments (y compris les segments vides) se trouvant sur les différentes branches. L'ensemble des postes candidats peut être généré d'une manière efficace à l'aide de certaines règles. Dans la figure I-12 nous présentons un exemple d'une telle génération.

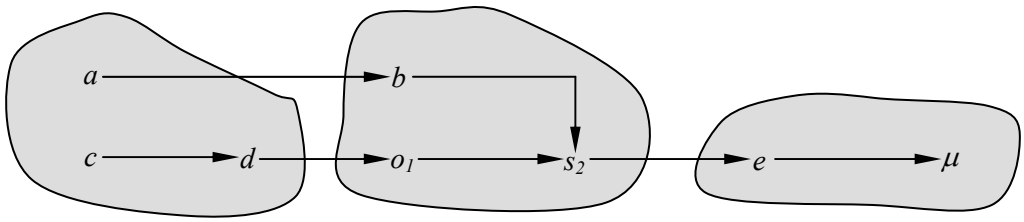


Fig. I-12. Détermination des postes candidats
au sein du graphe d'assemblage

I.2.5 Évaluation des méthodes de conception

Pour chacune des méthodes de conception des systèmes d'assemblage présentées auparavant nous pouvons mettre en évidence d'une part les avantages, et d'autre part, les inconvénients. Certaines de ces caractéristiques sont intrinsèques à la représentation des processus d'assemblage employée – telles que le graphe de précédence, la séquence d'assemblage et le graphe d'assemblage – tandis que d'autres résultent de l'algorithme utilisé. Par conséquent, une évaluation de ces méthodes doit se rapporter à ces deux points.

Les trois représentations des processus d'assemblage susmentionnées ont été conçues en vue de leur application pratique, plutôt que par souci de leur génération à partir du produit. Elles sont similaires du point de vue de leurs définitions et, comme suite, ont quelques points communs :

- elles sont plutôt compactes, malgré le fait que chacune est fréquemment éclatée en plusieurs graphes (ou arbres) ;
- elles permettent la discrimination, à l'aide de graphes distincts, des processus basés sur des tâches d'assemblage différentes ;
- elles mettent en évidence le parallélisme qui existe entre tâches (par exemple, cela peut être visualisé directement sur le graphe de précédence).

L'existence de ces points communs n'empêche pas qu'il existe de grandes divergences entre ces trois représentations. Évidemment, elles n'ont pas les mêmes applications dans l'assemblage.

Nous remarquons la supériorité des méthodes de conception basées sur les graphes de précédence. Cette affirmation s'appuie sur deux raisons principales détaillées ci-dessous.

- L'existence d'un *ordre partiel* aussi bien sur l'ensemble des tâches au sein d'un même poste candidat, que sur l'ensemble des postes candidats, laisse une grande flexibilité au niveau des processus d'assemblage. Au contraire, le système d'assemblage obtenu à partir d'un graphe d'assemblage ou d'une séquence est figé. Ce manque de flexibilité peut être partiellement contourné en adoptant un système de transfert en anneau, mais cela s'est avéré insuffisant par rapport aux besoins réels.
- Le nombre de graphes de précédence pour un produit donné n'est souvent pas grand, en tout cas nettement inférieur à celui des graphes ou des séquences d'assemblage. Il est, donc, préférable d'utiliser les graphes de précédence au lieu d'effectuer une exploitation systématique des graphes ou des séquences d'assemblage, qui est souvent infaisable à cause de leur nombre. Bien qu'il soit toujours possible d'effectuer une sélection préalable de ces deux représentations avant leur exploitation, cette approche n'est pas assez efficace en réalité, puisque toute sorte de sélection a priori introduit des éléments subjectifs.

Le point fort du graphe de précédence est que, une fois généré, il a une forme simple de représentation, directement applicable par l'esprit humain, ce qui en fait l'un des outils les plus puissants de la représentation des processus d'assemblage. Les pratiquants ont raison d'utiliser davantage cet outil aussi bien dans la conception, que dans le pilotage.

Néanmoins, l'utilisation des graphes de précédence n'est pas sans inconvénients. Le principal inconvénient réside dans la difficulté de sa génération, suite à l'ambiguïté de la définition des tâches et de la rupture assez abrupte avec le modèle du produit. À présent, pour la génération des graphes de précédence il n'existe encore pas de méthode systématique et suffisamment simple pour être applicable en pratique.

I.3 Conclusion

Au cours de ce chapitre nous avons d'abord introduit les **concepts de base** de l'assemblage, en soulignant ses principales particularités.

Nous avons présenté la **problématique de l'assemblage**, qui consiste essentiellement dans la conception d'un système pour assembler un produit supposé comme étant connu.

Nous avons présenté un état de l'art des **méthodes de conception des systèmes d'assemblage**. Ces méthodes reçoivent comme donnée d'entrée l'information sur un certain produit et doivent conduire à la structure du système capable d'assembler le produit en cause. Cette démarche engendre deux problèmes principaux : l'*analyse du produit*, ayant comme résultat le modèle du produit, et la *modélisation des processus d'assemblage* admissibles pour le produit donné.

Chacune des méthodes de conception que nous avons présentées a été accompagnée par le type de modèle du produit utilisé, aussi bien que par le modèle des processus d'assemblage employé.

Une conclusion importante s'est dégagée lors de l'évaluation des différentes méthodes de conception : *la supériorité des méthodes qui utilisent les **graphes de précédence** en tant que représentation des processus d'assemblage*. Cette supériorité est justifiée principalement par la "souplesse" du modèle, qui favorise la flexibilité du futur système d'assemblage.

Malheureusement, comme nous l'avons évoqué, la génération des graphes de précédence est un problème difficile.

Les assertions ci-dessus justifient la nécessité de l'étude des graphes de précédence pour l'assemblage, ce que nous ferons dans le chapitre II de ce travail. Les insuffisances des méthodes de génération existantes nous incite à proposer une nouvelle méthode, dont le fondement théorique est développé dans le chapitre III.

II PROPRIÉTÉS DES GRAPHES DE PRÉCÉDENCE

II.1 Introduction

II.1.1 Finalité

Nous savons que le graphe de précédence est un outil largement utilisé dans la conception des systèmes d'assemblage, surtout par la majorité des méthodes issues de l'équilibrage des lignes d'assemblage (ALB), qui utilisent ce type de graphes comme donnée d'entrée. C'est ainsi que la "fidélité" du modèle – en ce cas, le graphe de précédence - utilisé pour un processus d'assemblage connu se reflète dans la qualité du résultat de la conception. Nous avons vu que l'équilibrage est défini comme un problème de découpage en postes de travail, auquel on ajoute un critère d'optimum. Même si nous ne désirons pas réaliser l'équilibrage, l'importance du graphe de précédence en tant que modèle préliminaire est prouvée par la nécessité d'exploiter certaines de ses propriétés.

Deux inconvénients majeurs émergent pour le modèle de type graphe de précédence : l'inexistence de méthodes systématiques de génération et le fait que les tâches d'assemblage ne sont pas clairement définies du point de vue de la méthodologie d'assemblage. Dans tous les cas, la génération des graphes de précédence à partir du produit est encore faite de manière empirique par des experts. Cette démarche est assez compliquée lorsqu'il s'agit de la conception d'une nouvelle ligne d'assemblage ou d'un produit assez complexe. De plus, elle manque de garantie à l'égard de la qualité du résultat.

Le point de départ pour la détermination des graphes de précédence est le modèle du produit à assembler. Dans la démarche développée dans la thèse [CHEN 96] ce modèle est l'ensemble d'opérations d'assemblage. Cette méthodologie implique d'abord l'obtention de l'ensemble d'arbres d'assemblage équivalent aux opérations déduites, s'appuyant sur l'avantage que présentent les arbres d'assemblage de décrire correctement les tâches d'assemblage.

Il existe des procédures bien fondées de passage du modèle du produit au modèle des processus d'assemblage (par exemple, LEGA, logiciel conçu au L.A.B.). Un processus d'assemblage peut être modélisé comme un ensemble d'enchaînements partiellement ordonnés d'opérations d'assemblage. Une tâche d'assemblage englobe en général plusieurs opérations d'assemblage. Donc, chaque enchaînement d'opérations peut être vu comme enchaînement de tâches. Un tel enchaînement s'appelle *gamme d'assemblage*. Nous allons utiliser l'ensemble de gammes d'assemblage comme modèle des processus d'assemblage.

L'objectif de ce chapitre est l'étude des graphes de précédence pour l'assemblage, afin d'analyser le rôle joué par ce type de modèle dans la problématique générale de l'assemblage. Nous allons mettre en évidence **les propriétés des graphes de précédence**, leurs avantages et leurs faiblesses par rapport à d'autres modèles des processus d'assemblage. Pour mieux comprendre **la sémantique** et **la spécificité** des graphes de précédence, nous allons présenter différentes **méthodes de génération** des graphes de précédence à partir d'autres modèles des processus d'assemblage : opérations et arbres d'assemblage, graphes d'assemblage, gammes d'assemblage.

II.1.2 Synthèse des approches antérieures

Quelques travaux de recherche [KO 87][FROM 88][LEE 88][WEUL 89][DELC 90b] ont été menés pour l'obtention systématique des graphes de précédence. Même si les auteurs sont unanimes à l'égard de la définition formelle et de la sémantique du graphe de précédence, ces travaux n'ont pas pu pourtant contourner certaines difficultés et ambiguïtés de formalisation, ni définir des principes satisfaisants de génération.

Une des premières approches de la génération des graphes de précédence se trouve dans [WEUL 89], mais l'algorithme proposé manque de précision.

Dans [FROM 88] on déduit les contraintes de précédence à partir du modèle CAD du produit, en utilisant une *approche par désassemblage*. De l'ensemble de graphes de précédence satisfaisant cet ensemble de contraintes on ne retient que ceux qui sont acycliques. Leur nombre est ensuite réduit itérativement, en ajoutant trois types de contraintes supplémentaires portant sur le modèle du produit :

- de non pénétration ;
- de stabilité ;
- de minimisation du danger de la dissociation des parties déjà assemblées.

Le travail [DELC 90b] propose un *planificateur assisté par ordinateur* pour la génération des graphes de précédence entre opérations, afin d'obtenir ensuite les plans d'assemblage les plus parallèles. Les contraintes de précédence sont décrites par des faits en Prolog, mais certaines contraintes sont difficiles à décrire, ou bien leur description manque de rigueur, même si parfois les graphes de précédence générés sont corrects.

Dans [CHEN 96] on trouve une analyse exhaustive des principales insuffisances des travaux susmentionnés :

- *imprécision de la notion de "tâche d'assemblage"* – du fait de la représentation simplificatrice du graphe de précédence en tant qu'ensemble de tâches d'assemblage liées par des contraintes d'antériorité (cf. I.2.2.1) ; il existe deux niveaux de manifestation de cette ambiguïté : une tâche correspond en fait à un ensemble d'opérations, et cet ensemble diffère d'un graphe de précédence à un autre ;
- *passage direct du modèle du produit aux contraintes de précédence* – transformation directe des relations entre composants (contraintes d'assemblage) en relations binaires entre tâches d'assemblage (contraintes de précédence), ce qui est incorrect d'un point de vue méthodologique, lorsque les contraintes d'assemblage sont plus complexes que des simples relations binaires ;
- *méconnaissance sur la relation entre différents graphes de précédence* – situation présente dans de nombreux cas où les conséquences de l'utilisation de graphes différents ne sont plus connues (par exemple, dans la conception des systèmes d'assemblage).

Nous pouvons synthétiser les différentes approches listés ci-dessus par un schéma illustrant les variantes possibles de génération des graphes de précédence à partir d'autres modèles du processus d'assemblage (voir figure II-1).

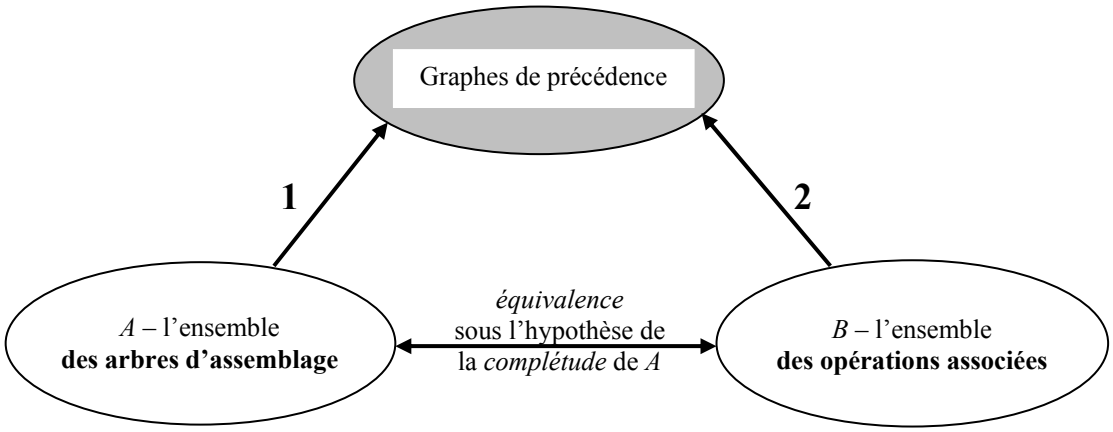


Fig. II-1. Relations entre graphes de précédence, arbres d'assemblage et opérations

La démarche notée par **1** a été proposée dans [HENR 92], où on introduit également la notion généralisée de "*hypergraphe de précédence*" pour décrire les contraintes disjonctives de précédence. Les deux chemins, **1** et **2**, sont en fait équivalents sous l'hypothèse de la complétude de l'ensemble d'arbres d'assemblages. Dans [CHEN 96] on montre que les deux ensembles *A* et *B* contiennent la même information sur un processus donné si la génération des arbres n'a pas subi de sélection a posteriori par évaluation. Le même ouvrage développe une méthode de génération qui évite les difficultés listées ci-dessus. La validité de cette méthode est prouvée sous une hypothèse raisonnable : que les contraintes matérielles ne soient pas prises en compte pendant l'élaboration des arbres d'assemblage. Néanmoins, cette méthode, dont les grandes lignes sont résumées ci-dessous, s'avère difficile et lourde à mettre en pratique.

Premièrement, le modèle opératoire du produit est englobé dans la formalisation du concept de "graphe de précédence". Ensuite, les tâches d'assemblage sont définies à partir des opérations d'assemblage. En ce point-là, on introduit la notion de "*tâche générique d'assemblage*", afin de désigner les opérations basées sur un même constituant secondaire ou une même tâche non géométrique. La base de la méthode est la relation entre les graphes de précédence et les arbres d'assemblage, qui est faite par l'intermédiaire des graphes d'assemblage. Pour des raisons qui portent sur la conception optimale d'un système d'assemblage, on introduit le concept de "*graphe de précédence maximal*" pour un ensemble de tels graphes. Ainsi on désigne le graphe capable d'assurer le maximum de flexibilité du système et on définit le but principal : l'obtention de tous les graphes de précédence maximaux pour un produit donné.

Quelques travaux plus récents – [DANL 99], [FOUD 99], [NAPH 99a, 99b], [LIT 00] – ont été consacrés à la détermination des graphes de précédence dans le cas multiproduit, c'est à dire pour une *famille de produits*.

Dans [DANL 99] les graphes de précédence sont obtenus à partir *d'une structure générique du produit* par l'intermédiaire d'une méthode heuristique.

L'idée de *produit générique* est adoptée aussi dans [FOUD 99], selon l'inspiration induite par le travail de Dini et Santochi [DINI 92] et par le formalisme de Fleury [FLEJ 93]. On introduit le concept de "*matrice d'interférence généralisée*" pour modéliser les interactions entre composants en chaque direction (positive) d'assemblage possible.

Notons cette matrice par I_{dt}, où *dt* désigne la direction d'assemblage. Par définition, $I_{dt}(i,j) = 1$ s'il existe une contrainte d'interférence généralisée entre les composants c_i et c_j (c'est à dire, c_j se trouve sur la trajectoire de désassemblage de c_i dans la direction *dt*). Il en résulte $I_{dt} = I_{-dt}^{T}$, où l'opérateur "^{T}" désigne la transposition matricielle et *–dt* est l'opposée de *dt*. Les lignes (respectivement les colonnes) vides correspondent aux composants qui peuvent être désassemblés dans la direction en cause (respectivement dans la direction opposée).

Le graphe de précédence est obtenu à travers une procédure d'actualisation des matrices d'interférence généralisée : ces matrices sont actualisées chaque fois quand il existe des composants qui peuvent être désassemblés. À noter que le composant de base est choisi par l'utilisateur, qui peut également proposer des sous-assemblages devant être considérés comme des parties solidaires.

Les travaux [NAPH 99a, 99b], après la démonstration de la NP-complétude du problème de l'identification d'une séquence (gamme) d'assemblage, développe une méthode systématique de génération de tous les graphes de précédence conformément à un *ensemble donné de contraintes de précédence*. La difficulté principale rencontrée est le traitement des *contraintes disjonctives*, c'est à dire celles de type "*a* précède *b* ***ou*** *a* précède *c*".

Les auteurs ont choisi de transformer les contraintes disjonctives en *contraintes conditionnelles*, en vertu de l'équivalence connue de l'algèbre booléenne : $(\neg p \vee q) \Leftrightarrow (p \Rightarrow q)$. Ainsi, elles peuvent être formalisées dans un graphe de décision, dont la partition engendre un ensemble de contraintes de précédence, équivalent à un graphe de précédence.

Malheureusement, à cause de la définition incorrecte du complément de "*a* précède *b*", la manière de transformation décrite est affectée d'inconsistance et peut conduire à des graphes de précédence cycliques. Le point faible de la procédure proposée est, donc, l'incapacité de déterminer la validité des graphes de précédence résultants.

Les deux derniers travaux cités ont inspiré l'approche proposée dans la thèse [LIT 00]. En ajoutant quelques changements à la méthode originale, pour éviter la génération des graphes de précédence invalides, l'auteur n'a pas cependant constaté des améliorations. Ce qui l'a conduit vers une stratégie plus proche de celle de Frommherz [FROM 88] : construire un *arbre de solutions possibles* selon l'ajout successif des contraintes de précédence. Chaque fois qu'on obtient un graphe invalide, la branche correspondante est éliminée.

La méthode proposée est intégrée dans la démarche d'obtention des gammes d'assemblage pour une famille de produits. On modélise cette famille par un ensembles d'*entités fonctionnelles*. À chaque entité fonctionnelle correspond plusieurs graphes de précédence, qui sont combinés pour donner une gamme d'assemblage de la famille.

Dans la même thèse on compare les différentes approches de génération des graphes de précédence du point de vue de la linéarité des gammes d'assemblage engendrées et de leur adaptation à des familles de produits.

II.2 Graphes de précédence comme modèle des processus d'assemblage

II.2.1 La sémantique du graphe de précédence

Dans la thèse [MÎNZ 95] on montre que la modélisation des processus d'assemblage se réalise en principe par l'intermédiaire des *diagrammes de transitions*. Cette approche correspond à la représentation des processus d'assemblage en tant que successions

d'événements. De ce point de vue, il existe un dualisme entre les diagrammes où les transitions sont réalisées par *les tâches d'assemblage* et celles où *les objets* réalisent les transitions. Ce qui conduit respectivement à deux représentations duales : les arbres d'assemblage et les graphes de précédence.

Dans la deuxième des variantes susmentionnées, un processus d'assemblage est décrit par un diagramme de transition, où les nœuds sont les tâches d'assemblage et les arcs correspondent au passage d'une tâche à une autre. Ce diagramme est en fait le graphe de précédence. Nous remarquons que la sémantique des tâches (des nœuds) est extérieure à la définition du graphe.

Donc, le graphe de précédence est *le graphe de la relation de précédence* définie sur l'ensemble *S* de toutes les tâches du processus. Si *a* et *b* ∈ *S*, alors la relation exprimée par "*a* **précède** *b*" signifie que la tâche *a* doit finir avant le commencement de la tâche *b*, pour des raisons issues du processus lui-même. La figure II-2 illustre un graphe de précédence.

La relation de précédence, définie de cette façon, est une *relation d'ordre partiel*. Les contraintes contenues dans le graphe de précédence expriment indirectement l'ordonnancement des tâches. Nous pouvons ainsi dire que le temps est implicitement présent. De plus, nous pouvons souligner un avantage du graphe de précédence : il engendre plusieurs plans d'assemblage (par exemple, il synthétise plusieurs arbres d'assemblage).

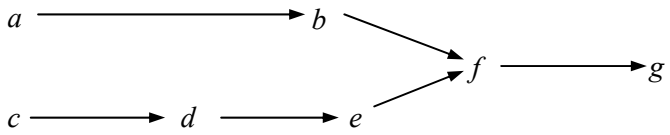


Fig. II-2. Exemple de graphe de précédence

Une observation importante doit être faite au sujet de la sémantique attachée aux nœuds du graphe. D'habitude, les étiquettes des nœuds sont les composants élémentaires utilisés. De cette façon, un nœud désigne la tâche qui assemble le composant correspondant avec le produit partiel réalisé jusqu'à ce moment-là. Par conséquence, on impose qu'il existe *un seul sous-ensemble* à un moment donné, auquel on ajoute un par un des composants élémentaires, ce qui correspond à une gamme (ou un arbre) d'assemblage linéaire.

En ce point-là nous remarquons un manque de rigueur du graphe de précédence, indiquée auparavant. Ainsi, un des deux constituants, qui définissent correctement une tâche d'assemblage, est défini ici d'une manière implicite. Par exemple, dans la figure II-2, la tâche *e* assemble le composant *f* avec l'objet formé par les composants *a*, *b*, *c*, *d* et *e*. Mais, en supposant qu'aucun sous-assemblage n'est pas utilisé, nous ne savons pas si le deuxième constituant de la tâche *b* est formé du composant *a*, ou bien des composants *a* et *c*, par exemple. Cette situation résulte de l'inexistence d'une relation de précédence entre *b* et *c*. Parmi les gammes (linéaires) représentées par ce graphe on rencontre, par exemple, $abcdefg$, $acbdefg$, $acdebfg$, $cdabefg$, etc.

Nous nous demandons quelles sont les étapes d'obtention des graphes de précédence pour un produit donné. Nous allons détailler ensuite ces étapes, qui utilisent des *procédures de conversion* entre les différents modèles du processus d'assemblage : arbres (gammes) d'assemblage, graphes d'assemblage et graphes de précédence.

II.2.2 Du modèle du produit aux opérations et arbres d'assemblage

La plupart des démarches de modélisation des processus d'assemblage passe obligatoirement par la procédure déjà bien connue de transformation du modèle du produit en modèle de type ensemble d'arbres (ou de gammes) d'assemblage. L'existence d'une telle

procédure est basée sur un logiciel élaboré en Prolog au L.A.B. [HENR 89]. Il s'appelle LEGA (Logiciel d'Élaboration des Gammes d'Assemblage) et il permet la détermination simultanée, en éludant les contraintes stratégiques, de l'ensemble d'opérations et d'arbres d'assemblage admissibles d'un produit connu comme prototype. L'explosion combinatoire impose que le produit ait moins d'une quinzaine de composants élémentaires.

L'utilisation de LEGA est invoquée dans la thèse [MÎNZ 95] – comme étape d'obtention des *graphes d'assemblage* - et reprise dans la thèse [CHEN 96], afin d'obtenir le modèle de type *graphe de précédence*. Dans le présent travail nous allons utiliser nous aussi les résultats fournis par LEGA (plus précisément, *les gammes d'assemblage*) comme données d'entrée pour notre méthode de génération des graphes de précédence.

LEGA peut être inséré dans une démarche globale donnée à la figure II-3 ([CHEN 96]).

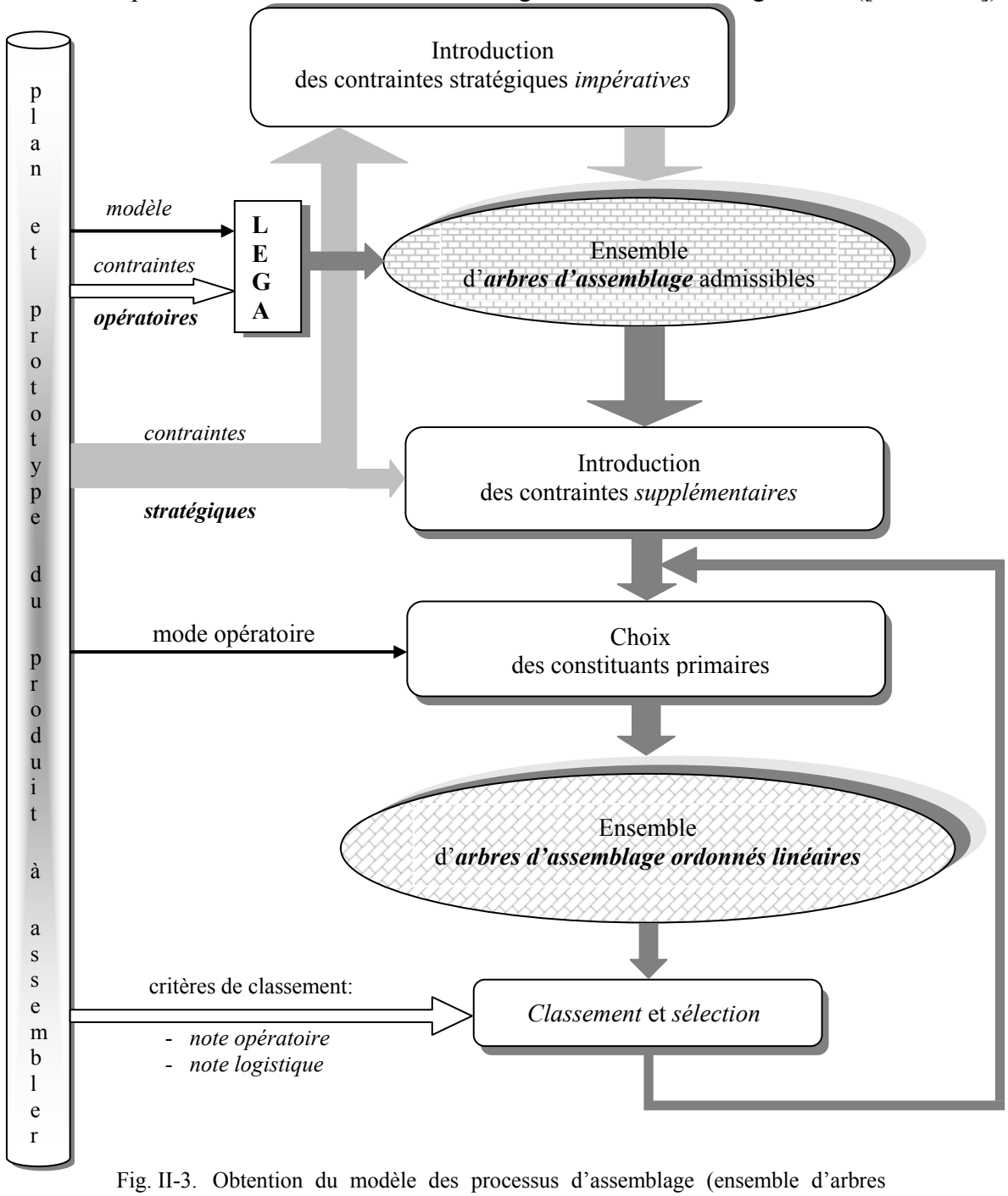


Fig. II-3. Obtention du modèle des processus d'assemblage (ensemble d'arbres d'assemblage) à partir du modèle du produit – méthode L.A.B.

Un **arbre d'assemblage ordonné** correspond aux opérations d'assemblage ordonnées. Une **opération d'assemblage ordonnée** est toute opération admissible complétée par la donnée de son constituant primaire – généralement représentée par la paire *ordonnée* (*S*, *e*) (voir I.2.4.2). Rappelons que le constituant *fixé* sur un posage est dit *primaire*, alors que celui qui est *déplacé* et monté sur le primaire est dit *secondaire*.

À chaque opération géométrique admissible correspond en principe *une seule* opération ordonnée, lorsqu'on conserve rarement toutes les deux possibilités. Évidemment, les opérations non géométriques sont déjà ordonnées. Quant aux opérations géométriques, l'obtention de leurs correspondantes ordonnées est faite par l'expert. Celui-ci construit sa décision sur trois éléments principaux :

- la maintenabilité du constituant secondaire ;
- le volume et le poids du constituant secondaire ;
- le type d'opération.

La notion de "**arbre d'assemblage ordonnée linéaire**" comporte une définition utilisant la notion de "**degré de linéarité**" associé à un arbre d'assemblage ordonné [CHEN 96]. Ce degré est le nombre de composants de base que l'arbre comporte. Un arbre d'assemblage ordonné de degré de linéarité égal à *1* est dit linéaire.

Cette définition est équivalente à celle de [MÎNZ 95] : tous les constituants secondaires sont des feuilles d'un arbre d'assemblage ordonné linéaire. Ce qui signifie que les constituants secondaires de toutes les opérations géométriques ordonnées ne peuvent être que des composants élémentaires. De telles opérations d'assemblage sont dites *simples*, par opposition à toutes les autres, qui sont dites *complexes*.

Nous remarquons qu'un arbre d'assemblage ordonné linéaire décrit un processus d'assemblage ne comportant que des opérations simples. Dans un tel processus on ajoute successivement les composants élémentaires à un composant choisi dès le début comme constituant de base.

La sélection par **contraintes stratégiques** peut intervenir *a priori* – avant l'introduction des contraintes opératoires – ou *a posteriori*, afin de réduire l'ensemble de solutions. Les contraintes stratégiques les plus souvent utilisées sont :

- les sous-assemblages / constituants de base / groupes de constituants imposés ;
- la restriction à des arbres linéaires.

Il est souhaitable d'imposer le dernier type de contraintes – rejet de toutes les opérations complexes – pour des raisons de simplicité de la mise en œuvre pratique.

L'évaluation est appliquée à l'ensemble d'arbres d'assemblage ordonnés linéaires en fonction de la *note opératoire* et de la *note logistique* de chaque arbre. Dans cette phase il s'agit d'estimer la difficulté opératoire et aussi la difficulté logistique des opérations. Dans [REMM 92] on trouve une étude approfondie de ce type de sélection.

Nous allons illustrer les étapes brièvement décrites ci-dessus sur un exemple simple.

Exemple II-1 :

Soit un produit didactique *P* formé de trois plaques *a*, *b* et *c* (voir figure II-4).

Les trois plaques sont solidarisées par deux ensembles de vis : un pour *a* et *b* et un pour *b* et *c*. Deux caractères non géométriques sont nécessaires pour modéliser les deux solidarisations : v_1 et v_2.

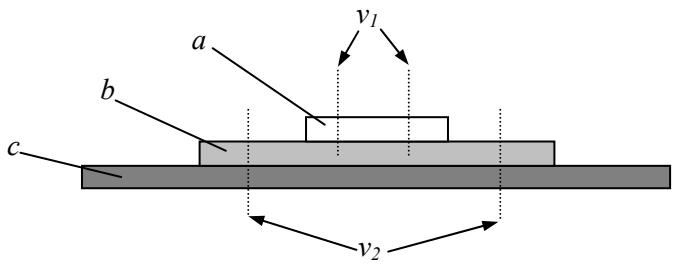


Fig. II-4. Produit didactique *P*

1. L'ensemble d'arbres d'assemblage *admissibles* est donné dans la figure II-5.

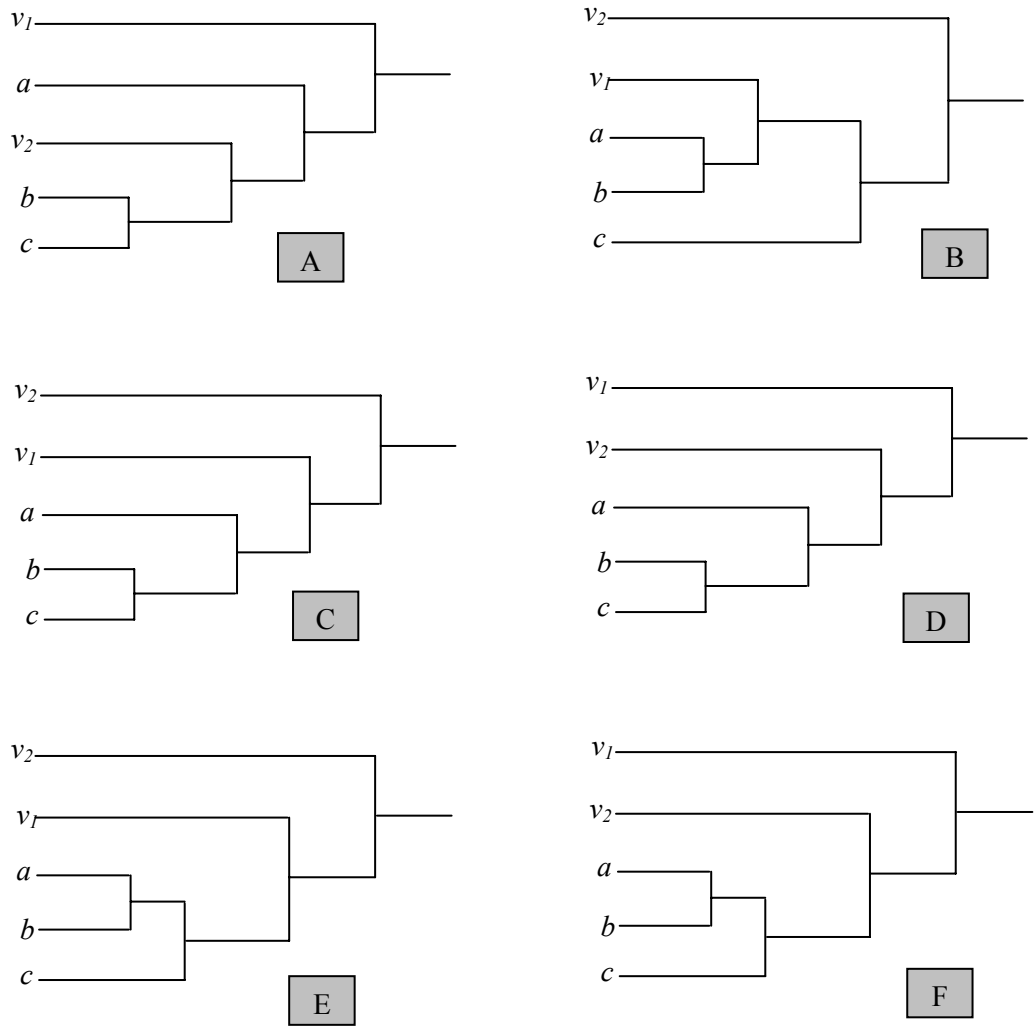


Fig. II-5. Ensemble d'arbres d'assemblage admissibles pour le produit *P*

2. L'ensemble d'arbres d'assemblage *ordonnés* est obtenu de l'ensemble admissible en ajoutant le choix des constituants primaires.

Comme nous l'avons montré auparavant, dans la plupart des cas, à chaque opération admissible correspond une opération ordonnée. Les opérations non géométriques – les deux solidarisations – sont implicitement ordonnées. Dans notre cas, il est raisonnable de choisir les constituants primaires des opérations géométriques (opérations d'assemblage proprement

dites) selon le critère du volume et/ou du poids. En regardant la figure II-4, nous pouvons supposer, par exemple, un ordre croissant des volumes des composants élémentaires : *a*, *b* et *c*. De plus, supposons que le sous-assemblage de *a* et *b* est moins volumineux que le composant *c*. Comme suite, nous déduisons six arbres d'assemblage ordonnés, notés selon leurs correspondants non ordonnés (voir figure II-6).

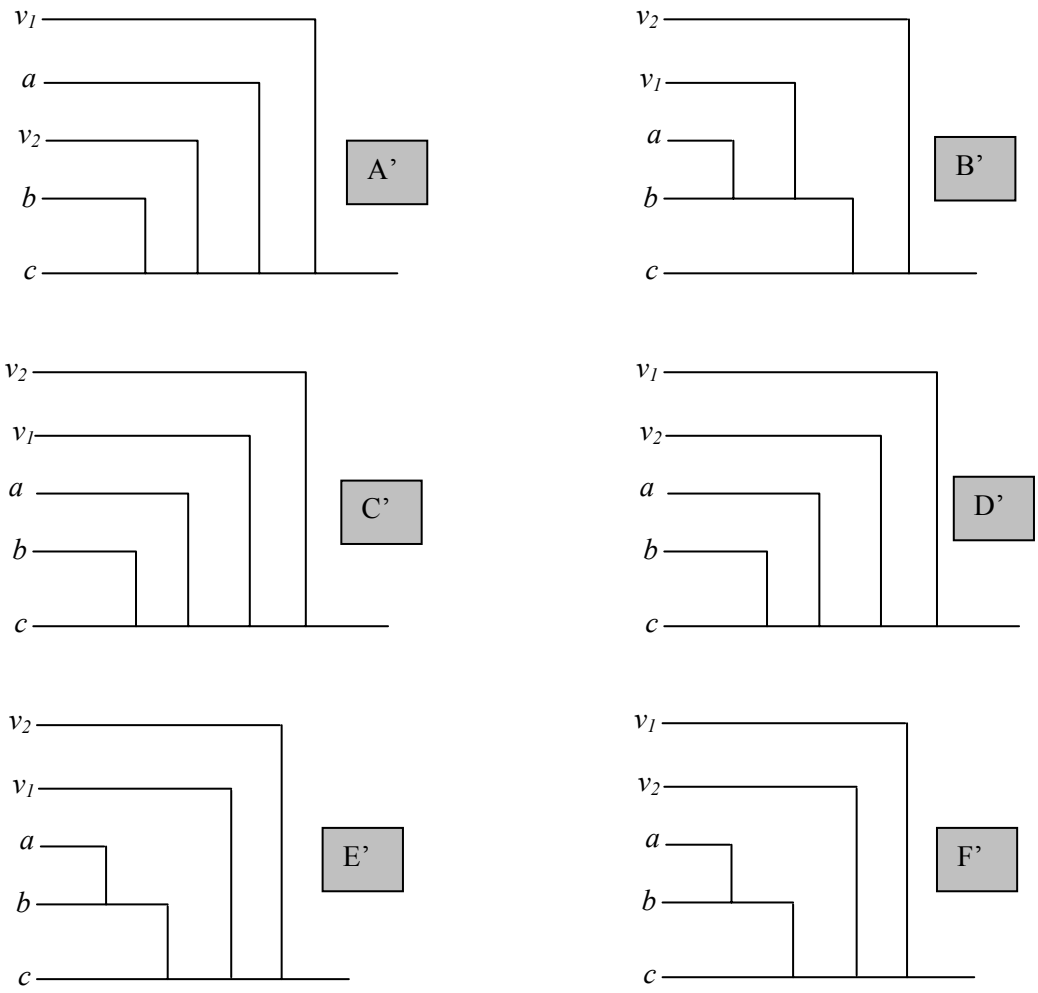


Fig. II-6. Ensemble d'arbres d'assemblage ordonnés pour le produit *P*

Notons que la construction d'un arbre d'assemblage *ordonné* à partir d'un arbre d'assemblage donné peut se faire d'une manière algorithmique [MÎNZ 95].

3. Nous pouvons observer qu'il existe trois arbres linéaires parmi les arbres d'assemblage ordonnés obtenus : A', C' et D'. L'ensemble d'arbres d'assemblage *ordonnés linéaires* est donné à la figure II-7.

4. Nous pouvons illustrer ensuite l'étape de *sélection par évaluation*. Il s'agit de l'étape où on tient compte de la note opératoire et de la note logistique de chaque arbre ordonné linéaire. Dans notre cas, il y a deux types d'opérations : le posage d'un composant sur un autre (ou sur un sous-assemblage) et l'opération de vissage. Trois changements d'outils sont nécessaires à la réalisation de l'arbre A', alors qu'il en faut seulement un pour la réalisation des arbres C' et D'. Une sélection par évaluation pourrait écarter l'arbre A', puisque sa note logistique est moins bonne.

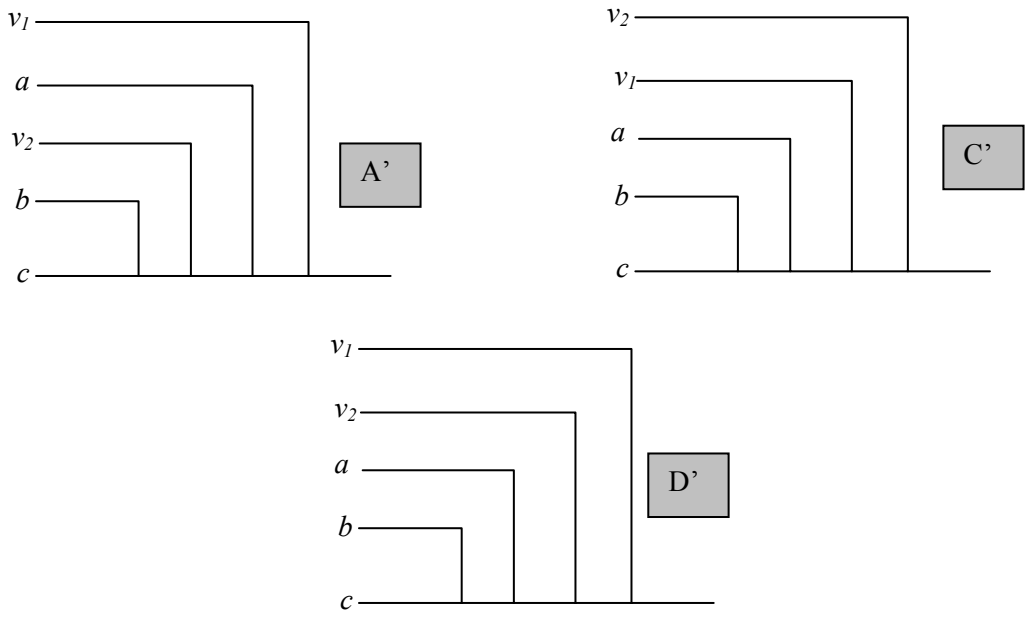


Fig. II-7. Ensemble d'arbres d'assemblage ordonnés linéaires du produit *P*

II.2.3 Graphes de précédence et graphes d'assemblage

Avec l'acquisition de la connaissance sur le constituant primaire de chaque opération, un arbre d'assemblage peut être transformé en un graphe d'assemblage. Dans [MÎNZ 95] on montre deux étapes intermédiaires de passage d'un arbre d'assemblage ordonné au graphe d'assemblage : l'obtention de l'arbre d'assemblage *orienté* et ensuite de l'arbre d'assemblage *orienté complété*.

Selon sa définition, un **graphe d'assemblage** est une anti-arborescence dont :
- la racine est la tâche de déchargement,
- les feuilles sont les tâches de chargement des composants de base,
- les nœuds intermédiaires sont les tâches d'assemblage qui désignent les opérations géométriques et non géométriques,

de sorte que les sous-assemblages résultant de chaque nœud constitue un constituant primaire ou secondaire de ses prédécesseurs.

Le graphe d'assemblage est essentiellement un graphe de précédence avec une structure particulière lui permettant de conserver les avantages de la représentation du processus d'assemblage par arbres. Cette conclusion émerge du fait que les nœuds du graphe ne représentent plus des objets, mais des tâches d'assemblage. Donc, les arcs expriment les relations de précédence entre tâches.

Afin que nous puissions donner un exemple de construction des graphes d'assemblage, nous faisons quelques conventions de représentation :
- toute tâche géométrique est représentée par le constituant secondaire ;
- toute tâche non géométrique est représentée par le caractère concerné ;
- toute tâche de chargement est représentée par le composant de base manipulé ;
- l'unique tâche de déchargement est symbolisée par μ.

(*Remarque* : On ne considère pas ici le cas où il y a plusieurs tâches de déchargement, l'extension à ce type de situation ne posant pas de difficultés particulières).

Exemple II-2 :

À la figure II-8 est montré l'ensemble de graphes d'assemblage du produit *P* de la figure II-4.

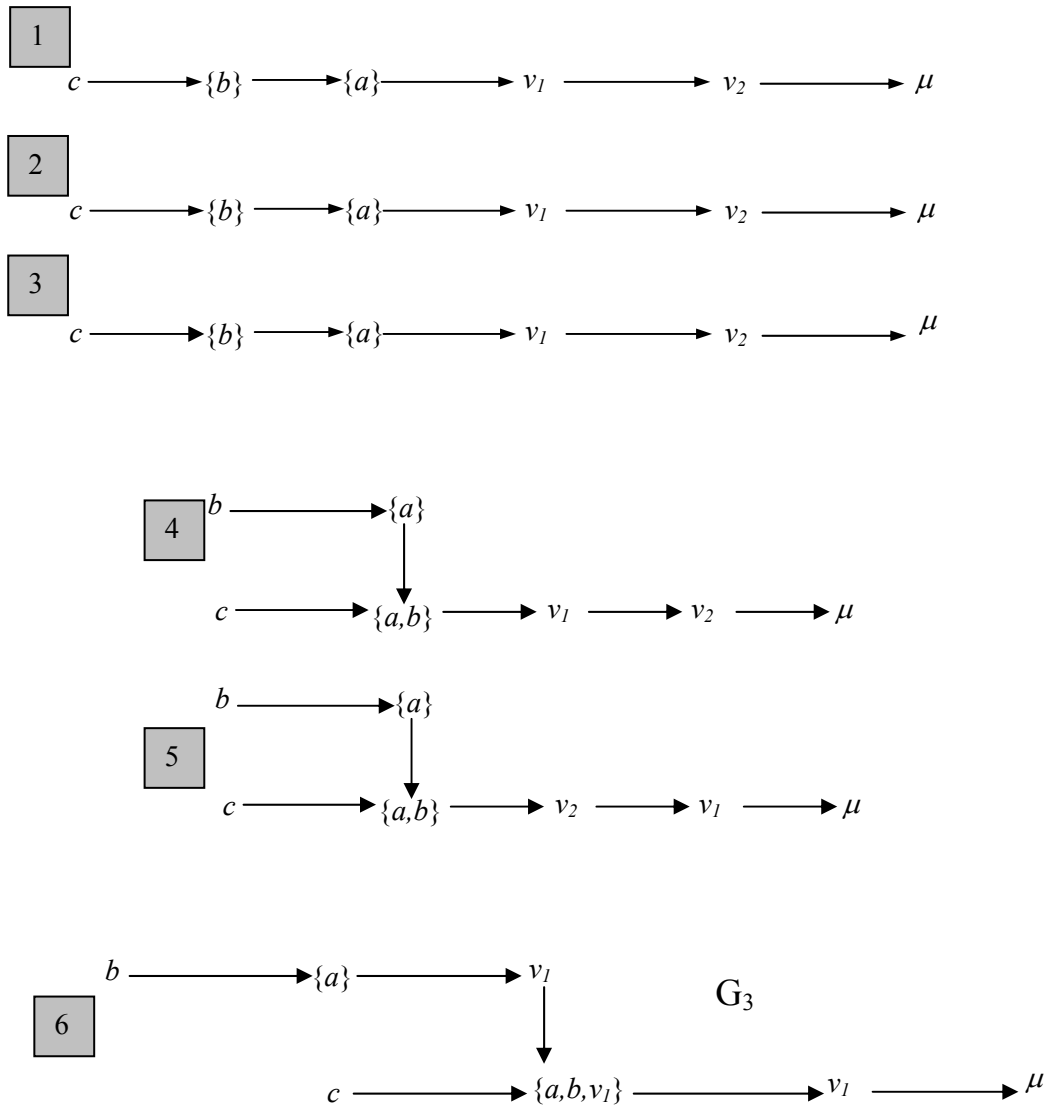


Fig. II-8. Graphes d'assemblage du produit *P*

Un graphe de précédence G est défini par un couple (*S*, *U*) où :

- *S* est un ensemble de tâches d'assemblage,
- *U* est un ensemble de contraintes de précédence définies sur *S*×*S*.

Graphiquement, un graphe de précédence G = (*S*, *U*) est représenté par son **graphe partiel** (*S*, *R*), où :

$$\forall t_1, t_2 \in S, (t_1, t_2) \in R \quad \Leftrightarrow \quad ((t_1, t_2) \in U) \wedge (\forall (t_1, t_3) \in U : (t_3, t_2) \notin U)$$

Donc, le graphe partiel est obtenu en éliminant les précédences "directes" s'il existe en parallèle *une chaîne* de précédences directes (nous reprenons ce point dans le chapitre suivant). Plus précisément, s'il existe un chemin en G de t_1 à t_2 et il existe aussi un arc, ce dernier sera éliminé. Nous pouvons déclarer cet arc comme arc redondant, puisque la précédence décrite par le chemin est plus "détaillée" (voir figure II-9).

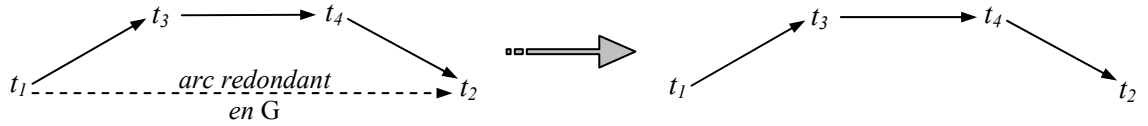


Fig. II-9. Principe de la représentation simplifiée d'un graphe de précédence

Un graphe de précédence est une **représentation collective** de plusieurs graphes d'assemblage, en ce sens qu'il impose *un ordre partiel moins restrictif* entre les tâches d'assemblage. Évidemment, les graphes d'assemblage sont définis sur le même ensemble de tâches d'assemblage. Par exemple, si deux arbres d'assemblage d'un produit quelconque imposent *des précédences contraires* pour deux tâches t_1 et t_2, alors le graphe de précédence correspondant n'aura *aucun arc* entre les deux nœuds t_1 et t_2. Dans ce sens-là, tout graphe d'assemblage est **inférieur** au graphe de précédence qui le représente. Autrement dit, un graphe d'assemblage est un graphe de précédence **minimal** [CHEN 96].

Exemple II-3 :

À partir des graphes d'assemblage du produit *P* (voir figure II-8) nous construisons trois graphes de précédence : G_1, qui engendre les graphes 1, 2 et 3, G_2, qui engendre les deux graphes suivants et G_3, qui est en fait le dernier graphe d'assemblage.

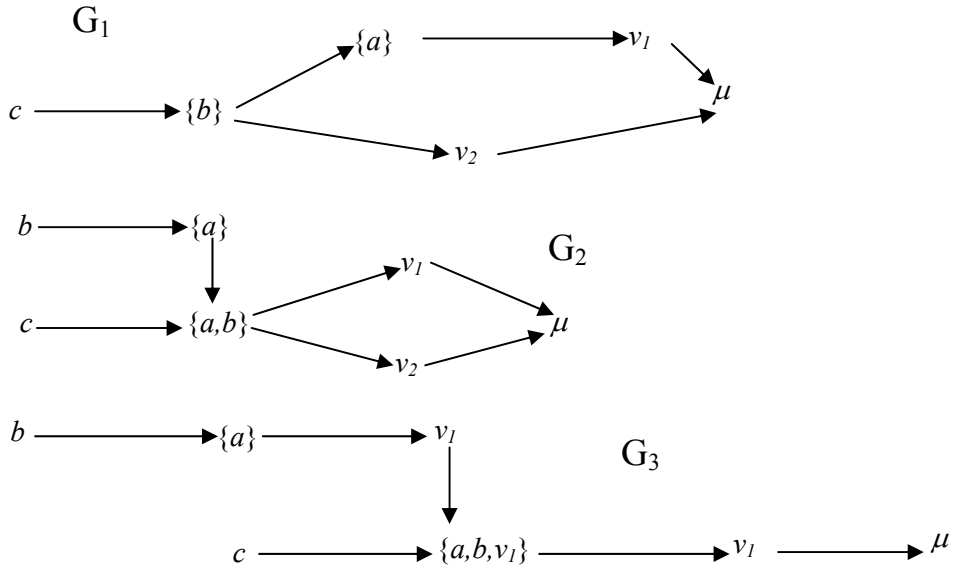


Fig. II-10. Graphes (partiels) de précédence pour le produit *P*

L'exemple ci-dessus montre qu'en général il existe plusieurs graphes de précédence pour un produit donné. Pour un ensemble de graphes de précédence on peut définir un graphe de précédence **maximal**, tel que tout autre graphe de l'ensemble lui est inférieur.

II.2.4 Spécificité des graphes de précédence

Nous allons faire quelques observations sur la sémantique de la relation d'ordre partiel représentée par le graphe de précédence. Lorsque certaines tâches ne sont pas liées par des contraintes de précédence, les relations entre elles peuvent être décrites par le **parallélisme** et la **permutation**.

Le parallélisme existe aussi au sein des graphes d'assemblage. Deux tâches sont dites parallèles si elles sont réalisées à partir de constituants indépendants. Ce qui se traduit au niveau du graphe de précédence par le fait qu'elles *n'ont pas des prédécesseurs en commun*.

Une *permutation* de tâches est un ensemble de tâches qui ont un *constituant de base commun* et qui peuvent être réalisées consécutivement selon *tous les ordres possibles* à partir du constituant commun. Le graphe de précédence permet de représenter les permutations, ce qui n'est pas possible pour les graphes d'assemblage.

Exemple II-4 :

Reprenons l'ensemble de graphes de précédence de la figure II-10. Dans la figure II-11 nous avons mis en évidence les tâches parallèles et les tâches permutables.

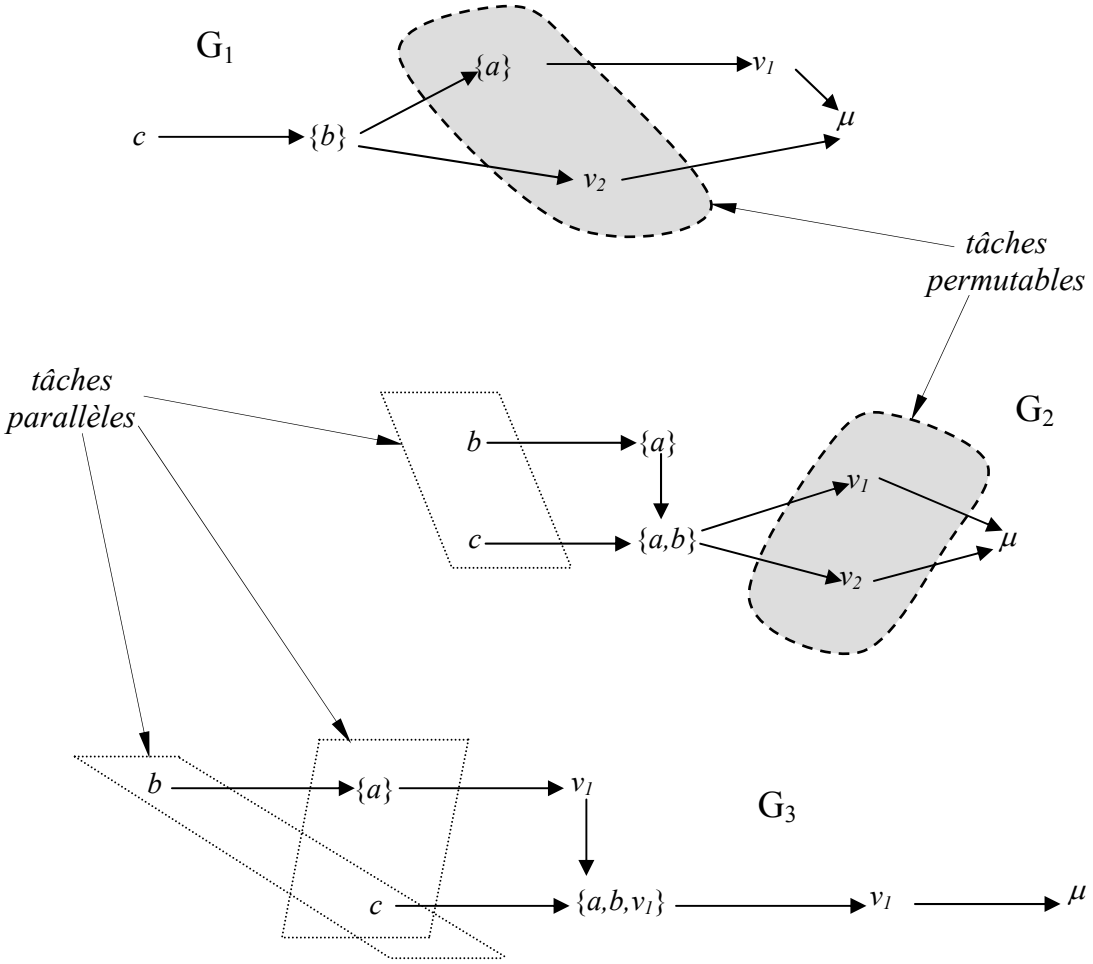


Fig. II-11. Le parallélisme et la permutation au sein des graphes de précédence

Dans le paragraphe II.3 nous introduisons la notion de **relation d'indifférence** entre les tâches d'assemblage, qui traite d'une manière unitaire aussi bien du parallélisme, que de la permutation.

Une autre caractéristique des graphes de précédence se manifeste comme une faiblesse par rapport aux graphes d'assemblage. Il s'agit de *l'impossibilité d'intégrer les tâches de changement d'orientation* au sein du graphe de précédence. Ce fait vient du manque d'information sur l'orientation des constituants primaires des tâches. L'intégration de ce type de tâches est toujours possible au niveau des graphes d'assemblage ordonnés, en conduisant à la définition des graphes d'assemblage orientés.

II.2.5 Graphes de précédence linéaires

Un graphe de précédence sera dit *linéaire* s'il ne comporte qu'*une seule tâche de chargement*. Ce qui se traduit par l'existence d'un nœud "de départ". Si un graphe de précédence est linéaire, alors tout graphe d'assemblage qu'il représente est *une chaîne*. Nous pouvons constater qu'à la figure II-10 seulement G_1 est linéaire.

Un graphe de précédence qui n'est pas linéaire peut être décomposé en sous-graphes linéaires selon les règles suivantes [CHEN 96] :

- chaque sous-graphe comporte une seule tâche de chargement ;
- les tâches de chaque sous-graphe ne contiennent pas le composant de base de ce sous-graphe.

De cette façon nous obtenons en fait l'ensemble des **sous-graphes de précédence linéarisés**, comme *solution unique* de cette décomposition, dont le principe est illustré à la figure II-12.

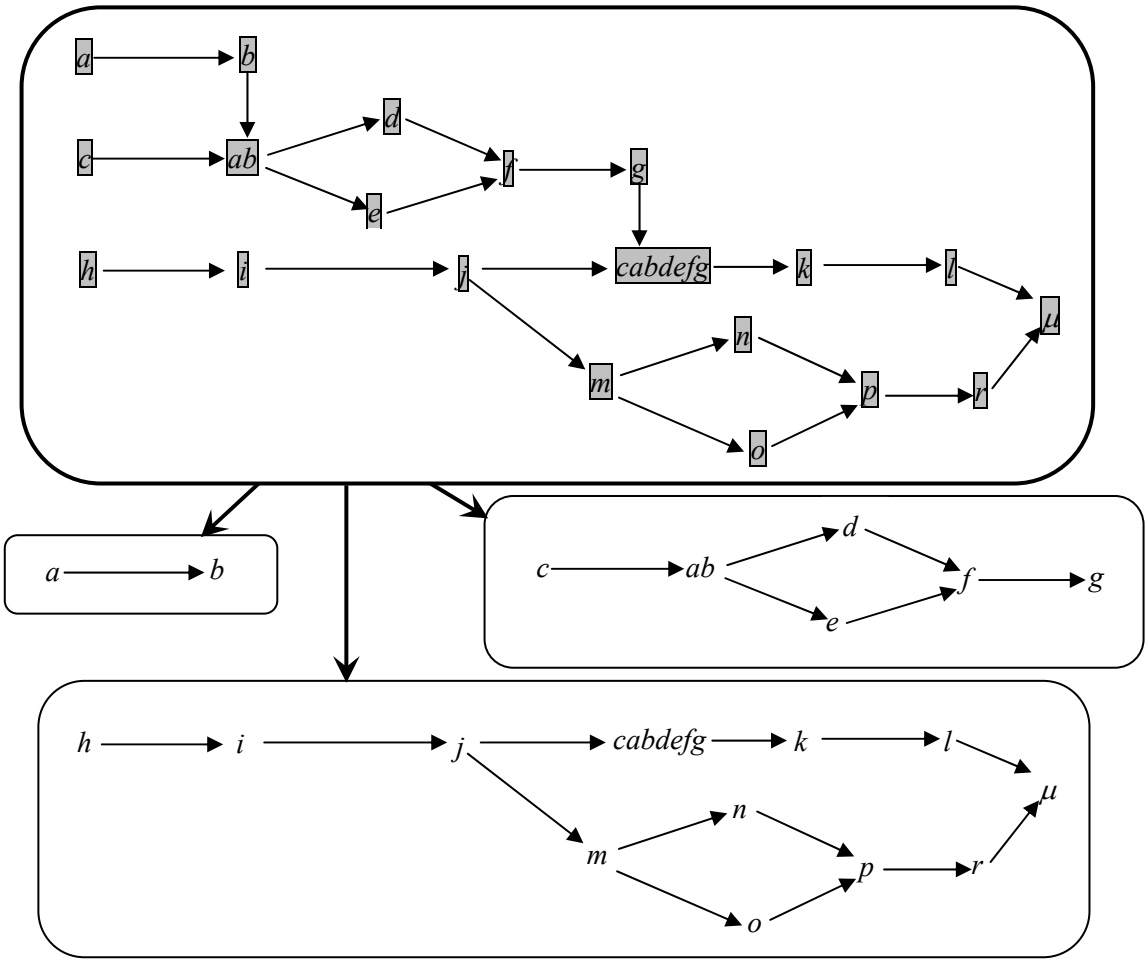


Fig. II-12. Exemple de décomposition d'un graphe de précédence en sous-graphes linéarisés

Nous donnons ensuite une présentation algorithmique de la procédure de linéarisation suggérée dans l'exemple antérieur.

L'algorithme reçoit comme donnée d'entrée un graphe de précédence quelconque et fournit comme résultat l'ensemble des sous-graphes linéarisés. Rappelons qu'une opération *simple* a comme constituant secondaire un composant élémentaire, toute autre opération étant appelée *complexe*.

Notons par *k* le nombre des nœuds du graphe. Nous adoptons aussi les conventions suivantes :

- un composant élémentaire est représenté par un *symbole* ; il en résulte que les nœuds du graphe – qui représentent les constituants secondaires des tâches (opérations) correspondantes – seront étiquetés par des *suites de symboles* formées :
  - d'*un seul* symbole dans le cas d'une opération *simple* ;
  - d'*au moins deux* symboles dans le cas d'une opération *complexe* ;
- la représentation informatique du graphe de précédence est *la matrice d'incidence des nœuds*, *M*, qui est une matrice carrée de dimension *k*, dont les éléments sont définis par la suite : $M_{x,y} = \begin{cases} 1, \text{si } x \text{ précède } y \\ 0, \text{sinon} \end{cases}$, où *x* et *y* sont des suites de symboles.

Les étapes de l'*algorithme de linéarisation d'un graphe de précédence* G (représenté sous forme de matrice d'incidence des nœuds, *M*) sont les suivantes :

1. Déterminer l'ensemble *CB* des symboles représentant les opérations de chargement, c'est à dire les composants de base des tâches du graphe (correspondant aux colonnes "vides" de la matrice *M*).
2. Déterminer l'ensemble *CC* des suites de symboles correspondant aux opérations complexes (ces suites commencent avec un des symboles déterminés en 1.)
3. Pour chaque suite α appartenant à *CC* :
   a) déterminer son dernier symbole, s_α, (le dernier composant élémentaire ajouté) ;
   b) effacer la précédence entre s_α et α : $M_{s_\alpha,\alpha} = 0$.
4. Diviser la matrice *M* en recherchant les lignes "vides", dont le nombre est égal à card(*CC*). L'ensemble des matrices résultantes, $\{M_i\}_{i=\overline{1,\text{card}(CC)}}$, contient les matrices d'incidence des nœuds des sous-graphes linéarisés.

Exemple II-5 :

Le graphe de précédence G de la figure II-13 a la matrice d'incidence des nœuds :

$M =$

	a	*b*	*c*	*ab*	*d*	*e*	*f*	*g*	μ
a	0	**1**	0	0	0	0	0	0	0
b	0	0	0	**1**	0	0	0	0	0
c	0	0	0	**1**	0	0	0	0	0
ab	0	0	0	0	**1**	**1**	0	0	0
d	0	0	0	0	0	0	**1**	0	0
e	0	0	0	0	0	0	**1**	0	0
f	0	0	0	0	0	0	0	**1**	0
g	0	0	0	0	0	0	0	0	**1**
μ	0	0	0	0	0	0	0	0	0

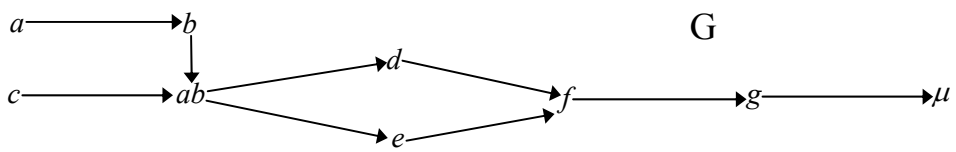


Fig. II-13. Graphe de précédence à linéariser

L'algorithme de linéarisation donne successivement :

$CB = \{a, c\}$, $CC = \{ab\}$,

la nouvelle forme de la matrice M étant :

		a	b	c	ab	d	e	f	g	μ
M_1	a	0	**1**	0	0	0	0	0	0	0
	b	0	0	0	**0**	0	0	0	0	0
	c	0	0	0	**1**	0	0	0	0	0
	ab	0	0	0	0	**1**	**1**	0	0	0
$M' =$	d	0	0	0	0	0	0	**1**	0	0
	e	0	0	0	0	0	0	**1**	0	0
	f	0	0	0	0	0	0	0	**1**	0
	g	0	0	0	0	0	0	0	0	**1**
M_2	μ	0	0	0	0	0	0	0	0	0

où nous avons marqué l'élément qui est devenu $0 - M'_{b,ab} = 0$ – lors de l'effacement de la précédence entre b et ab.

La deuxième ligne de la matrice est ainsi devenue "vide". Elle délimite deux matrices, M_1 et M_2, qui sont respectivement les matrices d'incidence des nœuds des sous-graphes linéarisés du graphe G, G_1 et G_2, montrés à la figure II-14.

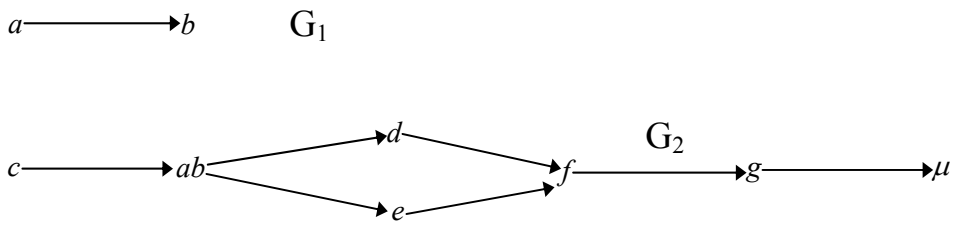


Fig. II-14. Sous-graphes linéarisés du graphe de précédence de la figure II-13

Structurellement, il existe deux différences mineures entre un sous-graphe de précédence linéarisé et un graphe de précédence linéaire :

- toute tâche constitutive (qui traduit une opération géométrique ou non géométrique) d'un graphe de précédence linéaire est simple, alors qu'elle peut être complexe au sein d'un sous-graphe de précédence linéarisé ;
- un graphe de précédence se termine toujours avec une tâche de déchargement, pendant que cette tâche peut être absente dans un sous-graphe de précédence linéarisé.

Donc, tout problème posé par un graphe de précédence quelconque peut être ramené à celui d'un ensemble de graphes linéaires. Nous dégageons une **conclusion** importante pour la suite de notre démarche. Le problème initial de génération des graphes de précédence à partir d'un ensemble de gammes d'assemblage – qui sont en fait des arbres d'assemblage linéaires, ou bien des graphes d'assemblage linéaires – peut être réduit à la génération des graphes de précédence à partir de chaque sous-ensemble de gammes qui commencent par *la même tâche de chargement*. Ces graphes de précédence sont linéaires.

Des détails sur ce passage seront donnés au paragraphe suivant.

La résolution du problème susmentionné – *l'objectif de notre travail* – présente un intérêt particulier, lorsqu'elle représente une des étapes de la transformation d'un ensemble quelconque d'arbres d'assemblage – issus, par exemple, de LEGA – en un graphe de précédence (voir figure II-15, où notre objectif a été symbolisé par des flèches grisées).

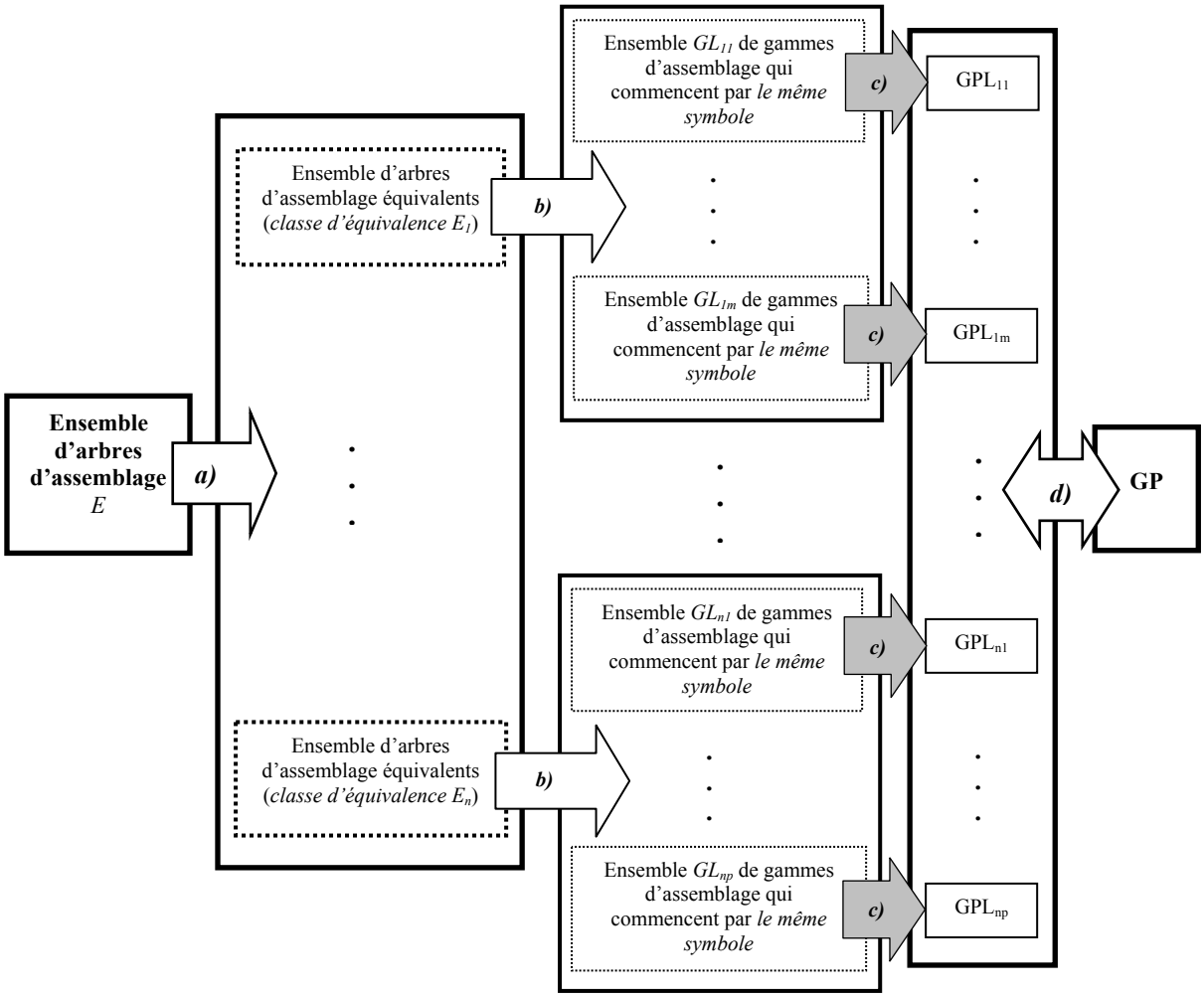


Fig. II-15. Étapes principales du passage d'un ensemble d'arbres d'assemblage à un graphe de précédence

L'étape ***a)*** concerne la *partition* d'un ensemble initial d'arbres d'assemblage, *E*, en des *classes d'équivalence* au sens de la relation d'équivalence entre deux arbres d'assemblage. Nous conservons la convention de la représentation des nœuds d'un arbre par les étiquettes des constituants secondaires des opérations correspondantes.

Deux **arbres d'assemblage** sont dits **équivalents** si :

- ils ont *le même ensemble de nœuds*
- *et* ils ont *le même ensemble de composants de base*.

La première condition revient à la coïncidence des ensembles des étiquettes des nœuds, tandis que la deuxième exprime simultanément l'égalité des nombres des *branches* et la coïncidence des premiers nœuds de ces branches.

Exemple II-6 :

Nous avons illustré dans la figure II-16 la relation d'équivalence sur un ensemble de quatre arbres d'assemblage. Ainsi :

- les arbres *1* et *2* sont équivalents, car les deux conditions requises sont simultanément remplies ;

- les arbres *2* et *3* ne sont pas équivalents, parce qu'ils n'ont pas le même ensemble d'étiquettes des nœuds ;
- les arbres *3* et *4* ne sont pas équivalents, parce qu'ils n'ont pas le même ensemble de composants de base ($\{a,x\}$, respectivement $\{a,c\}$) ;
- les arbres *2* et *4* ne sont pas équivalents, puisqu'aucune des conditions de la définition ci-dessus n'est remplie.

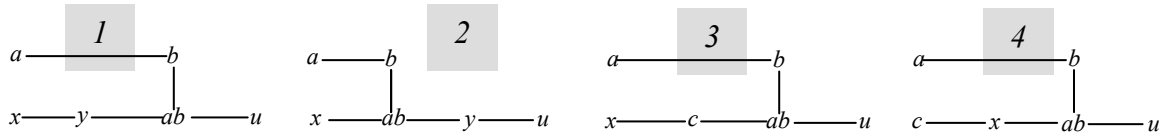


Fig. II-16. Relation d'équivalence sur un ensemble d'arbres d'assemblage

L'algorithme de partition s'appuie sur la vérification de l'équivalence entre deux arbres d'assemblage. Ce qui revient, en vertu de la définition ci-dessus, à employer un algorithme qui vérifie l'égalité des deux ensembles et ensuite, dans notre cas, l'égalité des suites de symboles. Du point de vue de la représentation informatique, les arbres d'assemblage et les graphes de précédence sont équivalents. Donc, nous pouvons choisir de représenter les arbres d'assemblage par leurs matrices d'incidence des nœuds. Tout en gardant les notations et les conventions faites auparavant, le schéma de l'algorithme de vérification de l'équivalence des deux arbres d'assemblage est donné ensuite.

L'algorithme reçoit comme données d'entrée les deux matrices d'incidence des nœuds, M_1 et M_2, chacune d'elles étant accompagnée par l'ensemble des étiquettes des nœuds S_i, $i=1,2$. L'algorithme fournit le résultat comme variable booléenne : "1" si les deux arbres sont équivalents, "0" sinon. Rappelons que la détermination des composants de base revient à la détermination des étiquettes des colonnes "vides" de la matrice d'incidence des nœuds.

Si $S_1=S_2$ *alors* #1. Déterminer l'ensemble CB_1 des étiquettes des colonnes vides de M_1
#2. Déterminer l'ensemble CB_2 des étiquettes des colonnes vides de M_2
Si $CB_1= CB_2$ *alors* "équivalence"
sinon "non équivalence"
sinon "non équivalence"

L'étape ***b)*** porte au niveau de chaque classe d'équivalence engendrée dans l'étape ***a)*** et a pour but l'obtention de la partition en des sous-ensembles d'arbres commençant par le même symbole (le même composant de base). Par conséquent, chaque arbre appartenant à une telle classe doit subir une *linéarisation*. L'algorithme employé dans ce but est pratiquement le même que celui déjà présenté au sujet de la linéarisation des graphes de précédence. En fait, au niveau des données reçues et des résultats fournis, il représente une particularisation de ce dernier : les arbres qui s'obtiennent sont des *ordres totaux* d'opérations, lorsque la linéarisation est dans ce cas un "découplage" des branches. C'est justement cela qui justifie l'appellation de "gammes" – qui sont linéaires – pour les arbres engendrés dans cette étape.

Dans l'étape ***c)*** chaque ensemble GL_{ij} de gammes produit un ou *plusieurs* graphes de précédence. Ces graphes sont linéaires. L'équivalence d'un ensemble de gammes linéaires à un seul graphe de précédence est garantie par une certaine propriété structurelle de cet ensemble (voir chapitre III).

L'étape ***d)*** est la *réciproque* de la linéarisation d'un graphe de précédence quelconque, problème que nous avons déjà traité. Les graphes de précédence linéaires de départ – notés génériquement par GPL dans la figure II-15 - sont en fait les sous-graphes de précédence linéarisés du graphe de précédence GP. Cette étape est symbolisée par une flèche bidirectionnelle, compte tenu de l'*unicité* de la décomposition d'un graphe de précédence en un ensemble de sous-graphes de précédence linéarisés (voir figure II-12 et exemple II-5).

II.3 D'un ensemble de gammes d'assemblage aux graphes de précédence

Dans ce paragraphe nous allons passer en revue les principaux aspects du problème de génération des graphes de précédence à partir d'un ensemble de gammes d'assemblage. Toute gamme d'assemblage peut être vue comme la traduction directe d'un arbre (ou graphe) d'assemblage linéaire. Si nous considérons que toute tâche est *simple*, au sens défini plus haut, alors elle peut être écrite comme *une suite de symboles*, qui sont les étiquettes des nœuds du graphe d'assemblage. Une telle suite représente en fait *une relation d'ordre total* sur l'ensemble des symboles.

II.3.1 Réduction à l'obtention des graphes de précédence linéaires

Le problème de la représentation d'un ensemble de gammes par graphes de précédence peut être ramené à celui de l'obtention des graphes de précédence linéaires. Nous décrivons ci-dessous le principe d'une telle réduction.

Tout d'abord, nous regroupons les gammes qui correspondent aux graphes qui ont *la même tâche de chargement*. Ainsi, nous obtenons des sous-ensembles qui commencent avec *le même symbole*. Cette décomposition conduit à *une partition* de l'ensemble initial de gammes en classes d'équivalence. Ensuite, nous devons *obtenir des graphes de précédence pour chaque sous-ensemble de gammes*. Ces graphes de précédence sont linéaires, conformément à ce que nous avons présenté dans le paragraphe antérieur.

Par exemple, selon le critère de la tâche de chargement commune, l'ensemble de gammes *X* représenté dans la figure II-17 contient quatre classes d'équivalence. La réunion des graphes de précédence déterminés pour chaque telle classe correspond à l'ensemble initial de gammes.

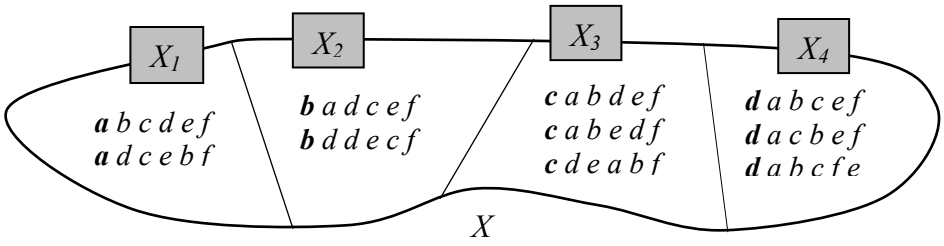


Fig. II-17. Partition d'un ensemble de gammes d'assemblage

Il en résulte que nous pouvons faire sans perte de généralité la supposition que tout ensemble de gammes est caractérisé par *un même premier symbole* ou, en d'autres termes, toutes les gammes d'assemblage commencent avec *la même tâche de chargement*. De cette façon, tout graphe de précédence qui représente cet ensemble sera *linéaire*.

II.3.2 Une première formulation du problème

Le problème que nous avons à résoudre a d'une part *un ensemble de gammes d'assemblage*, en tant que modèle du processus d'assemblage, et d'autre part *le graphe de précédence*, comme donnée d'entrée pour les méthodes de conception des systèmes d'assemblage. Notre objectif est de transférer la précision des modèles les plus connus des processus d'assemblage – gammes d'assemblage, arbres et graphes d'assemblage – au niveau des graphes de précédence, qui sont utilisés par la plupart des méthodes de conception des systèmes d'assemblage.

Nous remarquons que la construction du graphe de précédence se fait assez facilement, puisqu'il s'agit du **graphe de la relation de précédence sur l'ensemble des tâches d'assemblage**. Cette relation est une *relation d'ordre partiel*, qui s'obtient comme *l'intersection de tous les ordres totaux* introduits par chacune des gammes. La formulation mathématique de cette affirmation sera donnée dans le chapitre suivant. En ce point-là, nous nous contentons de présenter un exemple intuitif.

Exemple II-7 :

Construisons le graphe de précédence qui correspond à l'ensemble de gammes suivant :

$$\Omega = \begin{cases} a\ b\ c\ d\ e\ f\ g\ h\ i \\ a\ b\ d\ c\ e\ f\ h\ g\ i \\ a\ b\ c\ d\ e\ f\ h\ g\ i \end{cases}$$

Les trois gammes d'assemblage, représentées comme suites de symboles, reflètent trois possibilités d'ordonnancer les tâches d'assemblage pour un produit donné. L'expression synthétique de ces ordonnancements est une relation d'ordre partiel, dont le graphe G est représenté dans la figure ci-dessous et qui est le graphe de précédence associé à l'ensemble Ω.

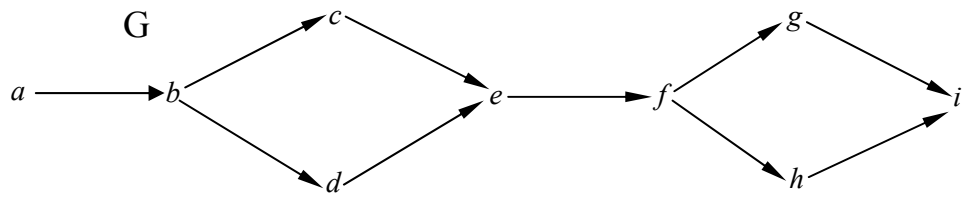


Fig. II-18. Le graphe de précédence G associé à l'ensemble Ω

Analysons maintenant le problème réciproque : *quelles sont les gammes représentées – ou "générées" – par un graphe de précédence donné* ? En d'autres termes, il s'agit de construire l'ensemble des séquences linéaires qui "*respectent*" le graphe. La notion d'une gamme qui respecte un graphe de précédence sera reprise et approfondie au cours du chapitre suivant. Pour l'instant, il est évident que le terme "respecter" concerne les contraintes (de précédence) imposées par le graphe.

Intuitivement, le graphe de l'exemple précédent génère les gammes suivantes :

a b c d e f g h i,

a b c d e f h g i,

a b d c e f g h i,

a b d c e f h g i.

Nous pouvons constater que le dernier ensemble contient une gamme supplémentaire par rapport à l'ensemble de gammes Ω. Ce qui signifie que *le graphe de précédence G génère plusieurs gammes* que les gammes à partir desquelles il a été déduit. Cette situation est généralement valable.

L'exemple qui suit illustre la situation contraire au dernier exemple : un ensemble quelconque de gammes peut correspondre à un graphe de précédence qui "génère" exactement les mêmes gammes que celles de départ.

Exemple II-8 :

l'ensemble ***donné*** *de gammes*

$$\Omega_1 = \begin{Bmatrix} a\,b\,c\,d\,e\,f; & a\,c\,b\,d\,e\,f; \\ a\,d\,b\,c\,e\,f; & a\,b\,d\,c\,e\,f; \\ a\,c\,d\,b\,e\,f; & a\,d\,c\,b\,e\,f; \\ a\,d\,c\,e\,b\,f; & a\,c\,d\,e\,b\,f. \end{Bmatrix}$$

$\Downarrow$

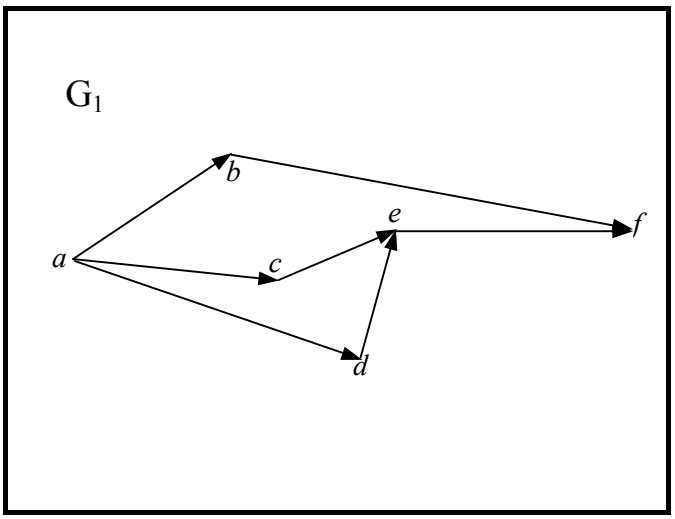


Fig. II-19. Le graphe de précédence qui représente l'ensemble de gammes Ω_1

$\Downarrow$

l'ensemble de gammes qui ***respectent*** G_1

$$\Theta_1 = \begin{Bmatrix} a\,b\,c\,d\,e\,f; & a\,c\,b\,d\,e\,f; \\ a\,d\,b\,c\,e\,f; & a\,b\,d\,c\,e\,f; \\ a\,c\,d\,b\,e\,f; & a\,d\,c\,b\,e\,f; \\ a\,d\,c\,e\,b\,f; & a\,c\,d\,e\,b\,f. \end{Bmatrix} = \Omega_1$$

Nous pouvons déjà conclure que, sous certaines conditions, il est possible d'avoir une superposition parfaite des deux ensembles de gammes, ce qui signifie que l'ensemble de départ – noté ci-dessus par Ω_1 – serait *équivalent* au graphe de précédence qui lui est associé.

Dans un premier temps, nous cherchons à détecter certaines propriétés d'un ensemble de gammes, telles qu'elles puissent garantir son équivalence avec un graphe de précédence. Il est assez clair que ces propriétés devraient caractériser *globalement* l'ensemble de gammes.

Lorsque nous décidons si un ensemble donné de gammes possède ou non telles propriétés, nous pouvons établir si l'ensemble peut être représenté par un seul graphe de précédence, ou bien par un ensemble de graphes de précédence. Évidemment, les mêmes propriétés nous permettraient de construire ces graphes de précédence.

Nous constatons que, en général, les gammes qui respectent un graphe de précédence G sont plusieurs que les gammes de l'ensemble donné, Ω. La question qui se pose est la suivante :

Est-ce que l'ensemble Ω *possède une propriété particulière lorsqu'il peut être représenté par un graphe* G, *respecté par seulement les gammes de* Ω ?

Nous allons montrer que la réponse est positive. La représentation biunivoque d'un ensemble de gammes Ω par un graphe de précédence G signifie la coïncidence de l'ensemble Ω avec l'ensemble de gammes qui respectent le graphe G. Notons ce dernier ensemble par Θ. L'ensemble initial Ω devrait, donc, avoir toutes les propriétés de Θ. Nous allons montrer que parmi ces propriétés se trouve la propriété structurelle qui garantit la représentation biunivoque que nous cherchons.

La propriété en cause sera nommée "**la propriété** Π" et elle reflète une certaine symétrie d'un ensemble de gammes d'assemblage, qui regarde les paires de symboles (de tâches) qui *ne sont pas en relation de précédence*. Nous dirons que tels symboles sont **indifférents**. Il s'agit, par exemple, de (*c,d*) et (*g,h*) dans l'exemple II-7, ou de (*b,c*), (*c,d*) et (*b,d*) dans l'exemple II-8.

Regardons l'exemple II-7. La gamme supplémentaire générée, qui manque à l'ensemble de départ, est *a b d c e f g h i*. Nous remarquons que cette gamme a la même structure que la gamme *a b c d e f g h i*, sauf que les symboles *c* et *d*, qui sont *indifférents* et se trouvent en *succession directe*, échangent leurs places. Nous dirons que les deux gammes sont **symétriques** par rapport à la paire (*c,d*). Remarquons aussi que la gamme en cause et la gamme *a b d c e f h g i* sont symétriques également, mais par rapport à la paire (*g,h*).

La conclusion est que dans l'ensemble de gammes qui respectent le graphe G se trouve **toutes les gammes symétriques** de la gamme *a b d c e f g h i*. Cette affirmation est valable pour **toute gamme** issue de G. Elle constitue l'essentiel de la propriété Π.

II.4 Conclusion

Dans ce chapitre nous avons présenté la problématique spécifique de la représentation des processus d'assemblage par des graphes de précédence. L'étude des graphes de précédence présente un intérêt spécial, compte tenu de leur utilisation par la plupart des méthodes de conception des systèmes d'assemblage. Ce que nous intéresse particulièrement est *l'utilisation des graphes de précédence par les méthodes d'équilibrage des systèmes d'assemblage*.

Pour comprendre la spécificité et la sémantique des graphes de précédence, nous avons fait l'encadrement de ce type de modèle dans l'ensemble des modèles les plus largement utilisés, par l'intermédiaire de quelques procédures de conversion de modèle, illustrées par des exemples suggestifs.

Nous avons mis en évidence les qualités des graphes de précédence, dont les plus importantes sont *la flexibilité* et *la compacité* de la représentation. Les graphes de précédence décrivent la relation de précédence entre les tâches d'assemblage. Nous avons montré comment les avantages d'autres modèles, plus "fidèles", peuvent être traduits dans une forme plus compacte, représentée par les graphes de précédence.

Nous avons également présenté, d'une manière plutôt descriptive, le problème sur lequel nous allons nous concentrer ensuite : *la représentation biunivoque d'un ensemble de gammes d'assemblage par un et un seul graphe de précédence*. Intuitivement, à l'aide de quelques exemples, nous avons anticipé l'existence d'une propriété globale de l'ensemble de gammes, qui lui permet d'être équivalent à un graphe de précédence. La démarche théorique concernant la mise en évidence de cette propriété sera détaillée au cours du chapitre suivant.

III DETERMINATION DES GRAPHES DE PRECEDENCE A PARTIR D'UN ENSEMBLE DE GAMMES D'ASSEMBLAGE

III.1 Objectif. Hypothèses de travail

L'objectif de ce chapitre est de fournir **une méthode systématique** d'obtention d'un ensemble de graphes de précédence qui représentent de façon biunivoque un ensemble donné de gammes d'assemblage. Cette méthode s'appuie sur la démonstration d'**une condition nécessaire et suffisante** de représentation d'un ensemble de gammes par *un seul* graphe de précédence : l'ensemble de gammes doit posséder une certaine propriété structurale globale. Les algorithmes présentés en fin de ce chapitre exploitent les résultats théoriques pour analyser, dans le cas général, l'équivalence entre un ensemble de gammes d'assemblage et un ensemble de graphes de précédence.

Avant d'introduire la formalisation mathématique du problème de détermination des graphes de précédence "équivalents" à un ensemble de gammes d'assemblage, nous devons présenter d'abord quelques hypothèses de travail sur les données d'entrée de notre démarche.

Nous considérons que **toute gamme d'assemblage** est la traduction directe d'**un arbre** (ou graphe) **d'assemblage linéaire**. De cette façon, toute gamme d'assemblage se présente sous la forme d'une **suite de symboles**, qui sont les étiquettes des nœuds du graphe d'assemblage. Ce qui signifie que les symboles utilisés désignent des tâches d'assemblage simples.

Nous voulons ramener notre problème à celui d'obtention des graphes de précédence linéaires. Dans ce but, nous allons faire une autre hypothèse qui porte sur un ensemble quelconque de gammes d'assemblage. Un tel ensemble peut être décomposé en sous-ensembles qui commencent avec *le même symbole*. La signification de cette opération est de regrouper les gammes qui correspondent aux graphes ayant *la même tâche de chargement*. Cette décomposition conduit à *une partition* de l'ensemble initial en classes d'équivalence. Ainsi, le problème initial est réduit à l'*obtention des graphes de précédence pour chaque sous-ensemble de gammes*. Comme nous l'avons vu dans le chapitre antérieur, ces graphes de précédence seront linéaires.

À titre d'illustration, considérons l'exemple suivant.

Exemple III-1 :

Nous voulons déterminer les graphes de précédence qui représentent l'ensemble de gammes *X* donné dans la figure III-1.

En regardant cette figure, nous voyons que trois classes d'équivalence s'obtiennent lors de la partition selon le critère du même premier symbole. Notre objectif est maintenant formé de *trois sous-objectifs* traités *indépendamment*, *mais de la même manière*: déterminer les graphes de précédence pour chaque sous-ensemble X_i, $i=1, 2, 3$. Nous pouvons dire que la décomposition a simplifié seulement la combinatoire du problème, sans réduire sa difficulté de principe.

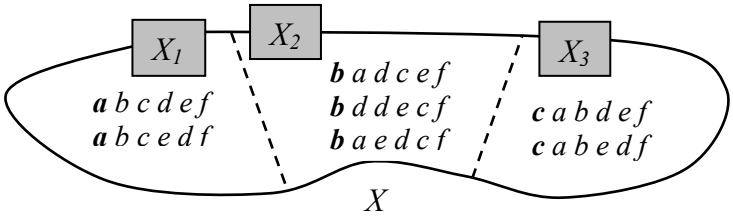


Fig. III-1. Partition d'un ensemble de gammes d'assemblage

Donc, sans perte de généralité, nous pouvons supposer dorénavant que tout ensemble de gammes est caractérisé par **un même premier symbole**. Il en résulte que, en ce qui suit, tout graphe de précédence peut être considéré comme **linéaire**.

III.2 Formulation du problème

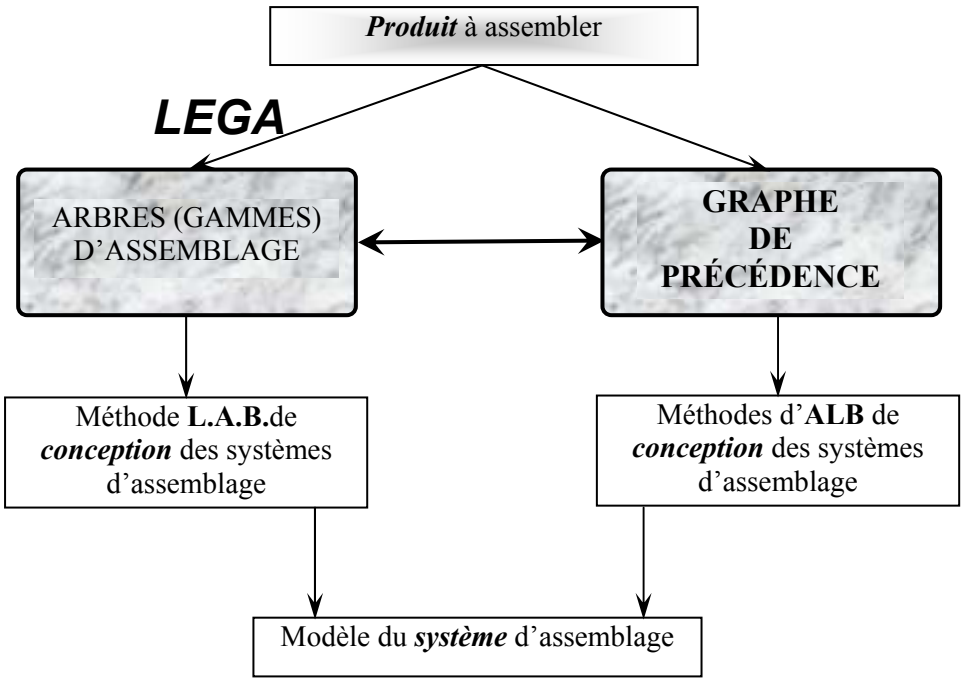


Fig. III-2. Justification du problème à résoudre du point de vue de la méthodologie de l'assemblage

Le problème que nous avons à résoudre regarde la liaison entre *un ensemble de gammes d'assemblage* en tant que modèle d'un processus d'assemblage et *le graphe de précédence* comme donnée d'entrée pour les méthodes de conception des systèmes d'assemblage. Notre objectif est de transférer la précision des modèles les plus connus des processus d'assemblage – gammes d'assemblage, arbres et graphes d'assemblage – au niveau des graphes de précédence, qui sont utilisés par la plupart des méthodes de conception des systèmes d'assemblage.

La figure III-2 présente une image suggestive des problèmes mentionnés ci-dessus. Nous allons nous intéresser au maillon qui manque de la chaîne de conception (dessiné par la flèche la plus grosse).

Formulation du problème

On donne :

Ω = un ensemble de gammes d'assemblage

On demande :

G = un graphe de précédence "équivalent" à Ω

Le but de ce chapitre est de formuler **une condition nécessaire et suffisante** pour que l'ensemble de gammes soit représenté **de façon biunivoque** par un graphe de précédence ou, autrement dit, de trouver le graphe de précédence "équivalent" à un ensemble de gammes dans un sens qui sera précisé ensuite. Ensuite, nous utilisons ces résultats théoriques pour **élaborer un algorithme de partage** d'un ensemble de gammes en sous-ensembles, tels que chacun d'eux soit équivalent à un seul graphe de précédence.

Intuitivement, les deux exemples qui suivent montrent qu'un ensemble quelconque de gammes peut correspondre à un graphe de précédence qui "génère" exactement les mêmes gammes que celles de départ (le premier exemple), mais, malheureusement, ce n'est pas toujours le cas (le deuxième exemple).

Exemple III-2 :

*l'ensemble **donné** de gammes*

$$\Omega_1 = \begin{Bmatrix} a\ b\ c\ d\ e\ f\ g\ h \\ a\ c\ b\ d\ e\ f\ g\ h \\ a\ b\ c\ d\ e\ g\ f\ h \\ a\ c\ b\ d\ e\ g\ f\ h \end{Bmatrix}$$

$\Downarrow$

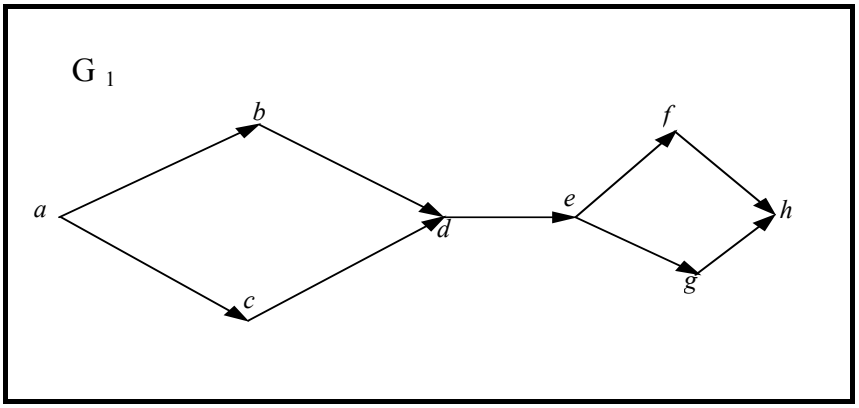


Fig. III-3. Le graphe de précédence qui représente l'ensemble de gammes Ω_1

$\Downarrow$

l'ensemble de gammes qui ***respectent*** G_1

$$\Theta_1 = \begin{Bmatrix} a\,b\,c\,d\,e\,f\,g\,h \\ a\,c\,b\,d\,e\,f\,g\,h \\ a\,b\,c\,d\,e\,g\,f\,h \\ a\,c\,b\,d\,e\,g\,f\,h \end{Bmatrix} = \Omega_1$$

Exemple III-3 :

l'ensemble ***donné*** *de gammes*

$$\Omega_2 = \begin{Bmatrix} a\,b\,c\,d\,e\,f \\ a\,c\,d\,e\,b\,f \\ a\,b\,d\,c\,e\,f \end{Bmatrix}$$

$\Downarrow$

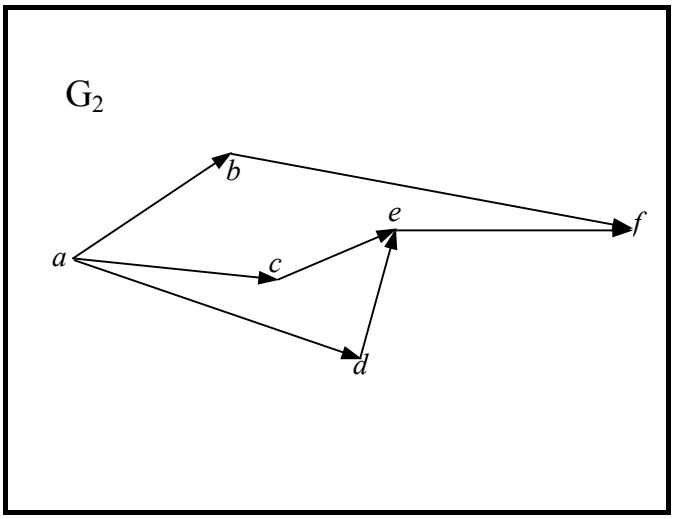


Fig. III-4. Le graphe de précédence qui représente l'ensemble de gammes Ω_2

$\Downarrow$

l'ensemble de gammes qui ***respectent*** G_2

$$\Theta_2 = \begin{Bmatrix} a\,b\,c\,d\,e\,f; & a\,c\,b\,d\,e\,f; \\ a\,d\,b\,c\,e\,f; & a\,b\,d\,c\,e\,f; \\ a\,c\,d\,b\,e\,f; & a\,d\,c\,b\,e\,f; \\ a\,d\,c\,e\,b\,f; & a\,c\,d\,e\,b\,f. \end{Bmatrix} \supset \Omega_2$$

Nous constatons qu'en général, le nombre des gammes qui respectent G est plus grand que le cardinal de l'ensemble donné de gammes.

Nous devons répondre à la question suivante: **l'ensemble Ω possède-t-il une propriété particulière lorsqu'il peut être représenté par un graphe** G, **respecté par seulement les gammes de Ω** ?

À partir de la description informelle du problème, nous allons le formuler mathématiquement. Comme nous l'avons montré, le problème peut être ramené à un travail sur les seules *gammes linéaires*, qui sont des suites de symboles. Quelques lemmes vont mettre en évidence les propriétés de ces suites, puis nous allons formuler les théorèmes et les résultats principaux. En dernier, un algorithme qui exploite ces résultats sera présenté.

III.3 Concepts et définitions. Définition formelle du problème

Dans ce qui suit, les gammes d'assemblage seront parfois appelées tout simplement "gammes". Pour définir le cadre conceptuel de notre problème nous adoptons les **notations** suivantes :

S - l'ensemble de symboles
card(S) = N
$a,b,c...$ - symboles de S
I=$\{1,2,...N\}$ - l'ensemble d'indices
M(S) - l'ensembles des suites formées de symboles de S : les gammes
$\Omega = \{ a_1 a_2 ... a_N \mid a_i \in S, i \in \mathrm{I} \}$ - un ensemble de gammes formées de symboles de S
($\Omega \in$ M(S))
ω - notation générique pour une gamme
card(Ω)=n

Chaque gamme ω_i, $i \in \{1,2,...n\}$, introduit *une relation d'ordre total* sur l'ensemble S des symboles, notée $O_i = S \times S$.

Un ensemble Ω de gammes introduit *une relation d'ordre partiel* sur l'ensemble S. Cette relation est l'intersection des relations d'ordre total introduites par toutes les gammes de Ω :

$$O_0 = \bigcap_{i=1}^{n} O_i$$

Le sous-ensemble du produit cartésien $S \times S$ qui contient les paires de symboles qui ne sont liés par aucune relation de précédence sera désigné par la **notation**

$$I = S \times S - \{ (a,b) \in S \times S \mid ((a,b) \in O_0) \vee ((b,a) \in O_0) \}$$

Nous allons donner deux définitions complémentaires.

Définition III-1 : relation de précédence "<"

On dit que le symbole *a précède* le symbole *b* et on écrit $a<b$ par rapport à l'ensemble Ω de gammes d'assemblage si $(a,b) \in O_0$.

Définition III-2 : relation d'indifférence "?"

On dit que deux symboles *a* et *b sont indifférents* et on écrit $a?b$ par rapport à l'ensemble Ω de gammes d'assemblage si $(a,b) \in I$.

Observations :

1) La relation de précédence est transitive :

$\forall\, x, y, z \in S$: $(x<y) \wedge (y<z) \Rightarrow (x<z)$;

elle n'est pas symétrique.

2) La relation d'indifférence n'est pas transitive, mais elle est symétrique :

$\forall$ x, y$\in$ S: (x?y) $\Rightarrow$ (y?x).

Définition III-3 : graphe de précédence

Le graphe de précédence associé à un ensemble de gammes d'assemblage est, par définition, **le graphe de la relation de précédence** entre les symboles des gammes d'assemblage (graphe orienté) :

G=$(S,\mathcal{O}_0)$, où :

S - l'ensemble des nœuds du graphe ;

$\mathcal{O}_0$ - l'ensemble des arcs du graphe en tant que paires ordonnées de nœuds.

Il résulte que le graphe de précédence s'obtient comme **l'intersection des ensembles des précédences déduites de toutes les gammes d'assemblage.** Donc, le graphe de précédence va contenir *des arcs redondants* qui correspondent aux *précédences qui émergent par transitivité.* C'est pourquoi le graphe de précédence sera représenté graphiquement sous la forme de son graphe partiel. Cette simplification se fait de la manière présentée dans le paragraphe II.2, comme illustré aussi dans l'exemple suivant.

Exemple III-4 :

Supposons que nous devons générer le graphe de précédence à partir de :

- l'ensemble de symboles $S = \{a,b,c,d,e,f\}$;

- l'ensemble de gammes d'assemblage $\Omega = \begin{cases} abcdef \\ acdebf \\ abdcef \end{cases}$.

L'algorithme de construction du graphe G a les étapes suivantes :

a) Chaque gamme introduit $\frac{\text{card}(S)\cdot(\text{card}(S)-1)}{2}$ précédences; en ce cas, il s'agit de quinze précédences introduites par chaque gamme.

$\omega_1 = abcdef$ contient les précédences $\mathcal{O}_1 = \begin{cases} a<b;\ b<c;\ c<d;\ d<e;\ e<f \\ a<c;\ b<d;\ c<e;\ d<f \\ a<d;\ b<e;\ c<f \\ a<e;\ b<f \\ a<f \end{cases}$;

$\omega_2 = acdebf$ contient les précédences $\mathcal{O}_2 = \begin{cases} a<c;\ c<d;\ d<e;\ e<b;\ b<f \\ a<d;\ c<e;\ d<b;\ e<f \\ a<e;\ c<b;\ d<f \\ a<b;\ c<f \\ a<f \end{cases}$;

$\omega_3 = abdcef$ contient les précédences $\mathcal{O}_3 = \begin{cases} a<b;\ b<d;\ d<c;\ c<e;\ e<f \\ a<d;\ b<c;\ d<e;\ c<f \\ a<c;\ b<e;\ d<f \\ a<e;\ b<f \\ a<f \end{cases}$.

b) L'intersection des ensembles de précédences introduites par toutes les gammes de Ω est : $O_0 = O_1 \cap O_2 \cap O_3 = \begin{Bmatrix} a < b; a<c; a<d; a<e; a<f \\ b < f \\ c < e; c<f \\ d<e; d<f \\ e<f \end{Bmatrix}$.

Donc, G=$(S, \circ_0)$, où :

$S = \{a,b,c,d,e,f\}$;

$\circ_0 = \{(a,b);(a,c);(a,d);(a,e);(a,f);(b,f);(c,e);(c,f);(d,e);(d,f);(e,f)\}$.

Le graphe de précédence est représenté dans la figure III-5a ci-dessous.

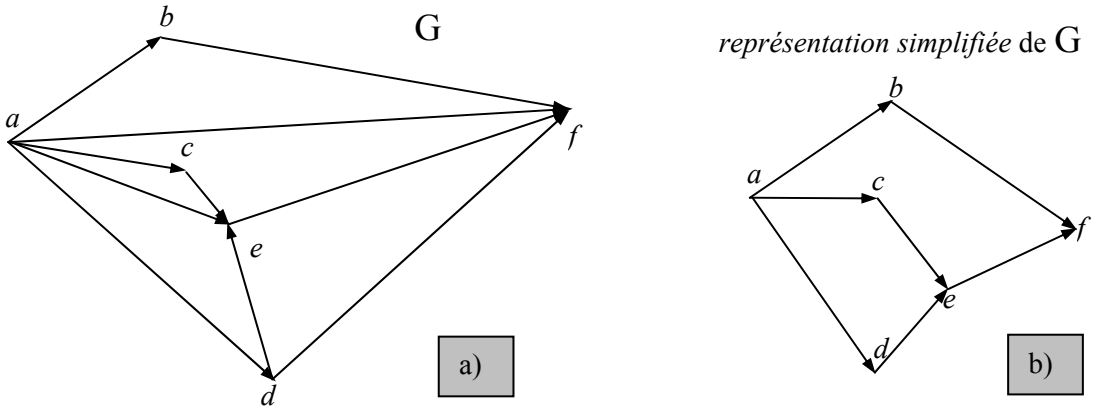


Fig. III-5. a) Le graphe de précédence issu de l'ensemble Ω et b) son graphe partiel

c) En éliminant de l'ensemble $\circ_0$ les précédences qui peuvent être déduites par transitivité, c'est à dire : *a<f; a<e; c<f; d<f,* on obtient la représentation simplifiée (le graphe partiel) du graphe de précédence. Dorénavant, c'est uniquement cette représentation qui va désigner le graphe de précédence (voir figure III-5b).

Observation :

La précédence entre deux nœuds est représentée au niveau du graphe de précédence **soit** par l'existence d'un **arc orienté**, **soit** par l'existence d'un **chemin** (chaîne d'arcs orientés) entre eux. **L'indifférence** entre deux nœuds est équivalente à la situation contraire.

Définition III-4 : gamme qui respecte un graphe de précédence

On dit qu'une gamme d'assemblage

$\omega = a_1 a_2 ... a_N$

respecte un graphe de précédence G si $\forall\, a_i, a_j \in S$ tels que *i<j,* alors **seulement** l'une des deux situations suivantes peut exister :

- il existe un **chemin** dans G entre a_i et a_j ;

- les deux symboles sont **indifférents** $(a_i ? a_j)$.

Dans ce qui suit, nous allons utiliser les **notations** suivantes :

Θ - ensemble de gammes d'assemblage qui respectent le graphe de précédence

ω,α,β,σ...minuscules grecques **:** suites finies de symboles de *S* (les gammes sont les suites formées de *N* symboles)

Nous devons faire une *remarque* concernant la notation d'une gamme : quand il s'agit de mettre en évidence certaines *suites* de symboles à l'intérieur d'une gamme, il sera utilisé l'opérateur de juxtaposition "." pour assurer la clarté d'écriture. Cet opérateur va être éludé quand nous voudrons écrire une gamme tout simplement en tant que juxtaposition de symboles.

Observation : Il est évident que, par la définition du graphe de précédence, toute gamme contenue dans l'ensemble Ω de départ va respecter ce graphe, donc toute gamme de Ω appartient à Θ, ce qui s'écrit : $\Omega \subseteq \Theta$ (en général, les gammes qui respectent le graphe G sont plus nombreuses que les gammes de départ).

III.4 Quelques propriétés des suites de symboles représentant les gammes

Ce sous-paragraphe contient les énoncés et les démonstrations des **lemmes** dont l'objet est l'ensemble de gammes qui respectent un graphe de précédence connu. Cet ensemble est noté par Θ et il possède quelques propriétés intéressantes.

Lemme L-III.1 :

Les gammes de Θ contiennent au moins une paire de symboles indifférents si et seulement si card(Θ)>*1*.

Justification :

Nécessité :

On sait qu'il existe des symboles indifférents, donc : $\bigcirc_0 \subset S \times S$. Soit $(a,b) \in S \times S - \bigcirc_0$. Il en résulte qu'il existe au moins deux gammes : une qui introduit la précédence *a*<*b*, une autre qui contient la précédence *b*<*a*. Donc, card(Θ)>*1*.

Suffisance :

On sait que card(Θ)>*1*. On utilise le raisonnement par l'absurde. On suppose qu'il n'existe pas des symboles indifférents. Parce que les paires de symboles indifférents appartiennent à l'ensemble `I`, il en résulte que cet ensemble est vide. Compte tenu de la notation de définition de l'ensemble `I` (voir le paragraphe antérieur), il s'ensuit que :

$$S \times S - \bigcirc_0 = \varnothing \Leftrightarrow \quad \bigcirc_0 = S \times S ;$$

par conséquent, la relation d'ordre partiel $\bigcirc_0$ est en fait une relation d'ordre total. Mais une telle relation ne peut être introduite que par un ensemble de gammes formé d'une seule gamme, ce qui signifie que card(Θ)=*1* – **contradiction** à l'hypothèse.

Nous avons obtenu une contradiction, donc la supposition faite est fausse.

En ce qui suit, pour les raisonnements qui ne font aucune mention spéciale, nous sous-entendons card(Θ)>*1*.

Définition III-5 : succession directe, succession indirecte

Soit ω une gamme d'assemblage formée de symboles d'un ensemble S et soient deux symboles a et b de S. Alors, on peut écrire ω sous la forme :

$\omega = \sigma_1.a.\sigma_2.b.\sigma_3$,

où σ_1, σ_2, σ_3 sont des suites de symboles. On dit que les deux symboles a et b apparaissent dans ω en **succession** :

- **indirecte** - si σ_2 *n'est pas vide* ;

- **directe** - si σ_2 est *vide*.

Lemme L-III.2 :

Soit ω une gamme de Θ et soient deux symboles indifférents qui apparaissent en succession indirecte dans ω. Alors **tout symbole** qui se trouve **"entre"** eux est **indifférent** au moins avec l'un des deux.

Démonstration :

En exprimant formellement l'énoncé du lemme, nous avons à montrer que :

$$\exists a,b \in S, a?b \text{ tels que:} \omega = \sigma_1.a.\sigma_2.b.\sigma_3 \Rightarrow \forall x \in \sigma_2 : (x?a) \vee (x?b) .$$

Nous raisonnons par l'absurde. Donc, supposons que :

$$\forall a,b \in S, a?b \text{ tels que:} \omega = \sigma_1.a.\sigma_2.b.\sigma_3 \Rightarrow \forall x \in \sigma_2 : \neg(x?a) \wedge \neg(x?b) .$$

Comme $\neg(x?a)$, donc x et a ne sont pas indifférents, alors x et a sont en relation de précédence. Le sens de la précédence est donné du fait que a se trouve avant x au sein de la gamme ω et, forcément, il y a de même dans toutes les gammes de Θ. Donc, $a<x$.

Un raisonnement analogue s'applique à partir de $\neg(x?b)$. Il vient que $x<b$.

Compte tenu de la transitivité de la relation de précédence, nous obtenons :

$$\left.\begin{matrix} a < x \\ x < b \end{matrix}\right\} \Rightarrow a < b ,$$

donc a et b sont en relation de précédence, en ***contradiction avec l'hypothèse*** (a et b sont indifférents) - et le lemme est ainsi montré.

Lemme L-III.3 :

Toute gamme d'assemblage de Θ contient au moins une paire de symboles indifférents en succession directe.

Démonstration:

Il faut montrer que :

$$\forall \omega \in \Theta, \exists a,b \in S, a?b \text{ tels que } \omega = \alpha.ab.\beta ,$$

où α et β sont des suites de symboles. Soit une gamme $\omega^* \in \Theta$ et soient a et b deux symboles indifférents qui apparaissent dans ω^* en succession indirecte :

$$\omega^* = \alpha.a.\sigma.b.\beta, a?b \text{ et } \sigma \text{ n'est pas vide} .$$

Conformément à L-II.2 appliqué pour les deux symboles a et b, il en résulte que :

$\forall x \in \sigma : (x?a) \vee (x?b)$.

Soit $x \in \sigma$. La gamme ω peut être écrite sous la forme : $\omega^* = \alpha.a.\sigma_1.x.\sigma_2.b.\beta$.

- si on suppose que $x?a$

- si σ_1 **est vide**, c'est à dire : $\omega^* = \alpha.a.x.\sigma_2.b.\beta$, il résulte que dans la gamme ω il existe *deux symboles indifférents* ($x?a$) *qui apparaissent en succession directe* - la démonstration est finie ;
- si σ_1 **n'est pas vide**, nous reprenons le raisonnement pour les symboles indifférents x et a, séparés par la chaîne σ_1 ;

Le nombre de tels raisonnements est limité, puisque le nombre de symboles de σ est fini. Finalement, en suivant une méthode inductive, nous arrivons à la conclusion que a doit se trouver en succession directe avec un symbole indifférent avec lui.

Le même raisonnement s'applique **si on suppose que** $x?b$.

La conclusion est qu'au moins l'un des deux symboles a ou b doit forcément se trouver en succession directe avec un autre symbole indifférent, ce qu'on a eu à montrer.

La démonstration du lemme L-III.3 est terminée.

Lemme L-III.4 :

> Pour chaque gamme de Θ et pour toute paire de symboles indifférents en succession directe de cette gamme, il existe une autre gamme dans Θ qui contient tous les symboles dans le même ordre, sauf les deux symboles qui échangent leurs places.

Justification :

Mathématiquement, l'énoncé du lemme s'écrit :

$\forall \omega \in \Theta, \forall a,b \in S, a?b$ *tels que* $\omega = \alpha.ab.\beta$

$\exists \omega' = \alpha.ba.\beta \in \Theta$.

Soit une gamme $\omega \in \Theta$. Comme suite au lemme L-III.3, ω contient (au moins) une paire de symboles indifférents en succession directe ; soit (x,y). Donc, en écrivant la gamme ω sous la forme $\omega = \alpha.xy.\beta$, nous devons montrer que la gamme $\omega' = \alpha.yx.\beta$ appartient aussi à Θ, c'est à dire qu'elle respecte le graphe de précédence G.

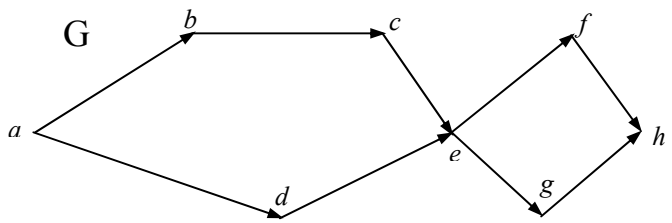
La **notation** "*XOR*" – inspirée de l'informatique – va désigner ensuite l'opérateur logique binaire "*ou exclusif*". À partir du fait que $\omega \in \Theta$ et de la définition d'une gamme qui respecte un graphe de précédence, nous faisons les raisonnements suivants :

- ω' ne contredit aucune relation de précédence entre x et y, parce qu'il n'en existe aucune ($x?y$) ;
- ω' garde les relations de précédence (s'il y en a) entre tous les symboles de la suite α et tous les symboles de la suite β, parce que les deux symboles n'échangent pas leurs places :
  $\forall z_1 \in \alpha, \forall z_2 \in \beta : (z_1 < z_2) XOR (z_1?z_2)$;
- ω' garde les relations de précédence (s'il y en a) entre x et tous les symboles de la suite α et entre y et tous les symboles de la suite α :
  $\forall z \in \alpha : ((z < x) XOR (z?x)) \wedge ((z < y) XOR (z?y))$;
- ω' garde les relations de précédence (s'il y en a) entre x et tous les symboles de la suite β et entre y et tous les symboles de la suite β :
  $\forall z \in \beta : ((x < z) XOR (x?z)) \wedge ((y < z) XOR (y?z))$.

C'est ainsi que nous avons montré que la gamme ω' respecte le graphe G.

Exemple III-5 :

Nous avons le graphe de précédence G et l'ensemble Θ associé.



$$\Leftrightarrow \quad \Omega = \begin{cases} abcdefgh \\ abcdegfh \\ abdcefgh \\ abdcegfh \\ adbcefgh \\ adbcegfh \end{cases}$$

Fig. III-6. Graphe de précédence et ensemble équivalent de gammes

L'ensemble de paires de symboles indifférents est : $\{\boldsymbol{b}?\boldsymbol{d}; \boldsymbol{c}?\boldsymbol{d}; \boldsymbol{f}?\boldsymbol{g}\}$. Soit, par exemple, la gamme $\omega = a\,\boldsymbol{b}\,\boldsymbol{d}\,c\,e\,\boldsymbol{f}\,\boldsymbol{g}\,h \in \Theta$. Remarquons que dans cette gamme toutes les paires de symboles indifférents apparaissent en succession directe :

- $\boldsymbol{b}$? $\boldsymbol{d}$ en succession directe $\Rightarrow \exists \omega' = a\,\boldsymbol{d}\,\boldsymbol{b}\,c\,e\,f\,g\,h \in \Theta$;
- $\boldsymbol{d}$? $\boldsymbol{c}$ en succession directe $\Rightarrow \exists \omega'' = a\,b\,\boldsymbol{c}\,\boldsymbol{d}\,e\,f\,g\,h \in \Theta$;
- $\boldsymbol{f}$? $\boldsymbol{g}$ en succession directe $\Rightarrow \exists \omega''' = a\,b\,d\,c\,e\,\boldsymbol{g}\,\boldsymbol{f}\,h \in \Theta$.

Rappelons la **notation**
$\mathtt{M}(S)$ – l'ensemble de gammes d'assemblage formées de symboles de S
Nous introduisons ensuite les **notations** :
X - un ensemble de gammes formées de symboles de S
$\mathtt{s} : \mathtt{M}(S) \rightarrow S \times S$, $\mathtt{s}(X) = \{(a,b) \mid a, b \in S, a?b\}$ - la fonction qui associe à l'ensemble de gammes X l'ensemble de paires des symboles indifférents

Définition III-6 : ***gamme symétrique***

Soit X un ensemble de gammes d'assemblage composées de symboles d'un ensemble S et soit une gamme ω de X. En écrivant ω sous la forme :

$\omega = \alpha.ab.\beta$,

où α et β sont des suites de symboles de S, (a,b) étant une paire de symboles de $\mathtt{s}(X)$, la gamme ω' qui s'écrit :

$\omega' = \alpha.ba.\beta$

s'appelle **la gamme symétrique de ω par rapport à la paire** (a,b).

Définition III-7 : ***propriété*** Π ***par rapport à un ensemble de paires de symboles***

Soit X un ensemble de gammes d'assemblage composées de symboles d'un ensemble S. Soit un ensemble P de paires de symboles de S. On dit que X **possède la propriété** Π **par rapport à l'ensemble** P si pour toute gamme de X et pour toute paire de P en succession directe, la gamme symétrique par rapport à la paire considérée appartient aussi à X :

$\forall \omega \in X, \forall (a,b) \in P \subset S \times S, \omega = \alpha.ab.\beta : \omega' = \alpha.ba.\beta \in X$.

Cas particulier (**propriété** Π) :

> Si $P = s(X)$, alors on dit tout simplement que X **possède la propriété** Π. Plus précisément, pour toute gamme de X et pour toute paire de $s(X)$ en succession directe dans X, la gamme symétrique par rapport à la paire considérée appartient aussi à X :
>
> $\forall \omega \in X, \forall (a,b) \in s(X), \omega = \alpha.ab.\beta : \omega' = \alpha.ba.\beta \in X$.

Compte tenu de la définition de la propriété Π, on peut exprimer différemment la propriété montrée dans le lemme L-III.4, concernant l'ensemble de gammes qui respectent un graphe de précédence : **Θ a la propriété Π.**

Nous allons utiliser la **notation** $P = s(\Omega)$. Nous faisons une *observation importante* sur la coïncidence des résultats de la fonction "s". Lorsque Ω et Θ définissent le même ordre partiel, alors, en appliquant la fonction "s" (qui associe à un ensemble de gammes d'assemblage l'ensemble des paires de symboles indifférents), aux ensembles Ω (de gammes de départ) et Θ, on obtient **le même résultat**: $P = s(\Omega) = s(\Theta)$, qui se confond à l'ensemble de nœuds indifférents du graphe G.

Lemme L-III.5 :

> Pour toute paire de symboles (nœuds) indifférents d'un graphe de précédence G il existe (au moins) une gamme appartenant à Θ qui les contient en succession directe.

Démonstration :

Nous avons à démontrer que :

$\forall a,b, a?b, \exists \omega \in \Theta$ *telle que* $\omega = \alpha.ab.\beta$.

Soit une paire quelconque de symboles indifférents (x,y) fixée. Nous écrivons toute paire de Θ comme :

$\omega = \alpha.x.\sigma.y.\beta$,

où la suite σ contient au plus *N-2* symboles (card(*S*)=*N*).

Nous utilisons l'**induction mathématique** pour montrer que l'existence d'une gamme ω telle que $\omega = \alpha.x.\sigma.y.\beta$, σ contenant k symboles, implique l'existence d'une gamme ω' telle que $\omega' = \alpha'.xy.\beta'$, pour tout $1 \le k \le N-2$.

P(*1*) : S'il existe une gamme ω de Θ telle que $\omega = \alpha.xay.\beta$ (la suite générique σ n'a qu'un symbole), alors il existe dans Θ la gamme $\omega' = \alpha'.xy.\beta'$.

Étant donné que *x*?*y*, on applique **le lemme L-III.2** à la gamme ω de Θ et aux symboles *x* et *y***,** en obtenant que *a*?*x* ou *a*?*y*. Nous utilisons la propriété Π de l'ensemble Θ et donc, la gamme symétrique de ω par rapport à la paire (a,x), $\omega' = \alpha.axy.\beta$ appartient aussi à Θ. Donc, la gamme cherchée a été trouvée : $\omega' = \alpha'.xy.\beta'$, où $\alpha' = \alpha x$ et $\beta' = \beta$; **P(*1*) est vraie**.

On suppose que **P(*k*) est vraie**. On doit montrer que **P(*k+1*)** est aussi vraie.

P(*k*) : S'il existe une gamme ω de Θ telle que $\omega = \alpha.x.\sigma.y.\beta$ et la suite σ a k symboles, alors il existe dans Θ la gamme $\omega' = \alpha'.xy.\beta'$.

P(*k+1*) : S'il existe une gamme ω de Θ telle que $\omega = \alpha.x.\sigma.y.\beta$ et la suite σ a $k+1$ symboles, alors il existe dans Θ la gamme $\omega' = \alpha'.xy.\beta'$.

Comme la suite σ contient $k+1$ symboles, nous détaillons la forme d'écriture de ω par la suite :

$$\omega = \alpha.x.\underbrace{s_1 s_2 \ldots s_{k+1}}_{\sigma}.y.\beta\ .$$

En appliquant **le lemme L-III.2** pour la gamme ω de Θ et pour les symboles indifférents x et y, il s'ensuit que :

$$\forall s_i,\ i=1\ldots k+1:\ (s_i\ ?\ x) \vee (s_i\ ?\ y)\ .$$

- Si $\forall s_i, i=1\ldots k+1:\ (s_i?y) \wedge \neg(s_i?x)$, c'est à dire que tous les symboles situés "entre" x et y sont indifférents <u>seulement avec y</u>, alors $\forall s_i,\ i=1\ldots k+1:\ (x<s_i)$; en utilisant **la propriété** Π **de** Θ, nous déduisons que la gamme symétrique de ω par rapport à la paire (s_{k+1},y) appartient à Θ :

$$\omega^* = \alpha.x.\underbrace{s_1 s_2 \ldots s_k}_{\sigma'}.y.s_{k+1}.\beta \in \Theta\ ,$$

où la suite σ' contient k symboles. Utilisons **P(*k*)** : nous avons montré qu'il existe la gamme ω^* dans Θ qui contient x et y séparés par une suite de k symboles. Donc, conformément à **P(*k*)**, il existe dans Θ une gamme au sein de laquelle les deux symboles se trouvent en succession directe.

- Si $\forall s_i,\ i=1\ldots k+1:\ (s_i?x) \wedge \neg(s_i?y)$, ce qui donne que $\forall s_i,\ i=1\ldots k+1:\ (s_i<y)$, on applique un raisonnement analogue en arrivant à la même conclusion.
- Le cas général est où tous les symboles $\{s_i\}_{i=1\ldots k}$ ne succèdent pas à x, alors il y en a quelques uns qui sont indifférents avec x ; soit s "le premier" symbole indifférent avec x, ce qui signifie que *tous les symboles situés entre x et s doivent forcément succéder à x*, alors nous pouvons détailler l'écriture de ω par la suite :

$$\omega = \alpha.x.\underbrace{s_1 s_2 \ldots s \ldots s_{k+1}}_{\sigma}.y.\beta, s\ ?\ x\ \text{ ; plus précisément :}$$

$$\omega = \alpha.x.\underbrace{\underbrace{s_1 s_2 \ldots s_p}_{\sigma'}.\boldsymbol{s}.\underbrace{s_{p+2} s_{p+3} \ldots s_{k+1}}_{\sigma''}}_{\sigma}.y.\beta\ ,\ \text{où}\ \forall s_i, i=1\ldots p\ (p \le k+1):\ x < s_i\ .$$

En appliquant **le lemme L-III.2** à la gamme ω de Θ et aux symboles indifférents x et s, il vient que forcément $\forall s_i, i=1\ldots p:\ s_i\ ?\ s$. Nous utilisons $p+1$ **fois la propriété** Π **de** Θ :

- la gamme symétrique de ω par rapport à la paire (s,s_p) appartient à Θ :

$$\omega^{(1)} = \alpha.x s_1 s_2 \ldots \boldsymbol{s s_p} s_{p+2} \ldots s_k y.\beta \in \Theta\ ;$$

- la gamme symétrique de $\omega^{(1)}$ par rapport à la paire (s,s_{p-1}) appartient à Θ :

$$\omega^{(2)} = \alpha.x s_1 s_2 \ldots \boldsymbol{s s_{p-1}} s_p s_{p+2} \ldots s_k y.\beta \in \Theta\ ;$$

...

- la gamme symétrique de $\omega^{(p-1)}$ par rapport à la paire (s,s_1) appartient à Θ :

$$\omega^{(p)} = \alpha.x \boldsymbol{s s_1} s_2 \ldots s_{p-1} s_p s_{p+2} \ldots s_k y.\beta \in \Theta\ ;$$

- la gamme symétrique de $\omega^{(p)}$ par rapport à la paire (s,x) appartient à Θ :

$$\omega^* = \alpha.\boldsymbol{s x} s_1 s_2 \ldots s_{p-1} s_p s_{p+2} \ldots s_k y.\beta \in \Theta\ .$$

Nous observons qu'en ce qui regarde ω, entre les symboles indifférents x et y se trouvent k symboles, donc, **conformément à P(*k*)**, il existe dans Θ une gamme dans laquelle les deux symboles se trouvent en succession directe.

Observation : Ce cas peut être caractérisé aussi par le fait que tous les symboles $\{s_i\}_{i=1...k}$ succèdent à *y* et, par conséquent, le même raisonnement peut être appliqué en choisissant *"le premier symbole à la gauche"* qui est indifférent avec *y*.

Nous avons montré que **P(*k+1*)** est vraie ; selon la méthode d'induction mathématique, il résulte que **P(*k*) est vraie pour tout** *k*, où $1 \leq k \leq N-2$.

Ensuite, parce que nous avons choisi une paire quelconque de symboles indifférents, **P(*k*)** reste valable pour toute telle paire. La démonstration du lemme L-III.5 est finie.

Définition III-8 : graphe d'indifférence

Soit un ensemble de gammes $\Omega \in \mathtt{M}(S)$. Par définition, **le graphe d'indifférence** – noté par G_I – associé à l'ensemble Ω est **le graphe non orienté de la relation d'indifférence** :

$$G_I = (S, \mathtt{s}(\Omega))$$

Afin d'obtenir le graphe d'indifférence à partir de l'ensemble de gammes, remarquons que l'ensemble des *nœuds* du graphe est l'ensemble des *symboles* des gammes et l'ensemble des *arcs* du graphe (nommés aussi **arcs d'indifférence**) est l'ensemble des *paires* non ordonnées *de symboles indifférents*.

Exemple III-6 :

La construction du graphe d'indifférence sera illustrée à partir de :

- l'ensemble de symboles $S = \{a,b,c,d,e,f,g,h\}$;
- l'ensemble de gammes d'assemblage $\Omega = \begin{Bmatrix} a\ b\ c\ d\ e\ f\ g\ h \\ a\ c\ d\ b\ e\ f\ g\ h \\ a\ c\ b\ e\ d\ f\ g\ h \\ a\ b\ d\ c\ e\ g\ f\ h \\ a\ c\ b\ d\ e\ f\ g\ h \end{Bmatrix}$.

Selon sa définition, le graphe d'indifférence est désigné par $G_I = (S, \mathtt{s}(\Omega))$, donc il faut mettre en évidence l'ensemble de paires de symboles indifférents qui émergent de Ω ; en ce cas-là, on obtient : $P = \mathtt{s}(\Omega) = \{(b,c);(b,d);(d,c);(d,e);(f,g)\}$. Le graphe d'indifférence G_I associé à Ω est montré dans la figure III-7.

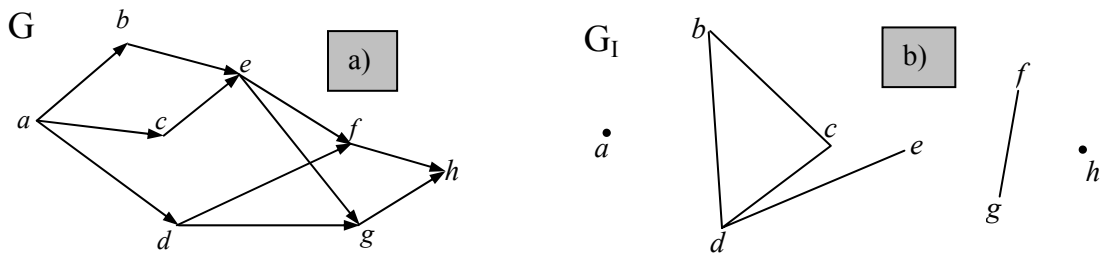


Fig. III-7. a) Le graphe de précédence et b) le graphe d'indifférence pour l'exemple III-6

Il est important de remarquer que :

G_I **s'organise en composantes connexes**.

Une *chaîne* dans le graphe G_I est une succession d'arcs d'indifférence où, autrement dit, une succession de paires d'indifférents ; évidemment, cette succession n'est pas transitive. Compte tenu de la définition d'une composante connexe d'un graphe, nous voyons qu'un ensemble de nœuds de G_I, désigné par la notation générique *C*, est une composante connexe si entre toute paire de nœuds *x* et *y* il existe une chaîne dans le graphe G_I. Ainsi, du point de vue formel, une composante connexe du graphe d'indifférence peut être caractérisée par :

$$C = \left\{ x \in S \mid \forall y \in C, \exists \left\{ z_j \right\}_{j=1...p} \in C: (x ? z_1) \wedge (z_1 ? z_2) \wedge ... (z_p ? y) \right\}.$$

Ensuite, nous allons utiliser les **notations** suivantes :

m - le nombre de composants connexes de G_I
$\{C_i\}_{i=1...m}$ - les composants connexes de G_I

Évidemment, les composantes connexes réalisent ***une partition*** du graphe G_I concernant aussi bien ses nœuds, que ses arcs :

$$\begin{cases} \bigcup_{i=1}^{m} C_i = X_I \\ C_i \cap C_j = \varnothing, \forall i, j = 1...m \end{cases}$$

Dans l'exemple III-6 : $m=4$, $C_1=\{a\}$, $C_2=\{b,c,d,e\}$, $C_3=\{f,g\}$, $C_4=\{h\}$.

Conséquence

Tous deux nœuds qui appartiennent respectivement à deux composantes connexes différentes quelconques se trouvent en relation de précédence.

Justification:

Ce que nous devons justifier s'écrit mathématiquement :

$\forall C,C', \forall a \in C, \forall a \in C': (a < a')XOR(a' < a)$.

Soient deux composantes connexes fixées, *C* et *C'***.** Soit *a* un nœud quelconque fixé de *C* et soit *a'* un nœud quelconque fixé de *C'***.**

Supposons que *a* et *a'* ne se trouvent pas en relation de précédence. Il en résulte que *a?a'*, d'où vient que $C \cap C' \neq \varnothing$, *contradiction*. Donc, la supposition faite est fausse.

Observations :

1) Comme la relation d'indifférence n'est pas transitive, *dans la même composante connexe il existe en général des nœuds qui se trouvent en relation de précédence*. Nous en rapportant à l'exemple ci-dessus, dans la composante C_2 il existe **les précédences** : *b<e; c<e*.

La chaîne mise en évidence entre tous deux nœuds d'une composante connexe de G_I, qui est définitionnelle pour la connexité, contient *plus qu'un arc <u>seulement</u> s'il s'agit de symboles en relation de précédence*. Dans l'exemple précédent, entre les symboles *b* et *e* où entre *c* et *e* il existe des chaînes de deux arcs, alors qu'entre toute paire de nœuds indifférents il existe des chaînes formées d'un seul arc.

2) Dans [CHEN 96] on trouve la définition de la **permutation maximale**, qui correspond à un ensemble de tâches d'assemblage qui peuvent être exécutées selon tous les ordres possibles. Au niveau du graphe de précédence, cette situation se traduit dans un ensemble de nœuds qui ne contient **aucune précédence**, alors qu'au niveau du graphe d'indifférence une permutation maximale représente une **composante connexe complète**.

Lemme L-III.6 :

Soit le graphe de précédence G et le graphe d'indifférence G_I associés au même ensemble de gammes. Pour toute gamme qui respecte G, pour toute paire de symboles qui appartiennent à une même composante connexe de G_I, tous les symboles situés entre eux dans la gamme appartiennent aussi à cette composante.

Démonstration:

Il faut prouver que :

$$\forall \omega \in \Theta,\ \forall C_i,\ i = 1...m,\ \forall a,b \in C_i:\ \omega = \alpha.a.\sigma.b.\beta \ \text{ et } \forall z \in \sigma\ z \in C_i .$$

Soit une gamme $\omega \in \Theta$ quelconque fixée, soit une composante connexe C_i quelconque fixée et soit deux symboles a et b appartenant à C_i. Nous écrivons la gamme ω en mettant en évidence les symboles a et b :

$$\omega = \alpha.a.\sigma.b.\beta .$$

Selon la définition d'une gamme qui respecte un graphe de précédence, en ce qui concerne la relation entre a et b, il peut exister deux cas distincts: soit $a<b$, soit $a?b$.

1) Si $\underline{a?b}$, alors nous appliquons le lemme L-III.2 à la gamme ω et à la paire de symboles indifférents (a,b) :

$$\forall x \in \sigma: (x?a) \vee (x?b) .$$

Selon la définition d'une composante connexe du graphe d'indifférence, il s'ensuit que $\forall x \in \sigma:\ x \in C_i$, ce que nous avons eu à montrer.

2) Si $\underline{a<b}$, nous utilisons toujours la même définition, donc :

$$\exists \left\{z_i^j\right\}_{j=1...p} \in C_i: (a?z_i^1) \wedge (z_i^1?z_i^2) \wedge ...(z_i^p?b) .$$

Nous utilisons le résultat du premier point, donc :

- tous les symboles entre a et z_i^1 appartiennent à C_i ;
- tous les symboles entre z_i^1 et z_i^2 appartiennent à C_i ;

...

- tous les symboles entre z_i^p et b appartiennent à C_i.

Quel que soit l'emplacement des symboles $\left\{z_i^j\right\}_{j=1...p}$ dans la gamme ω, la suite σ est "remplie" par des symboles de C_i. Ce que nous avons voulu montrer.

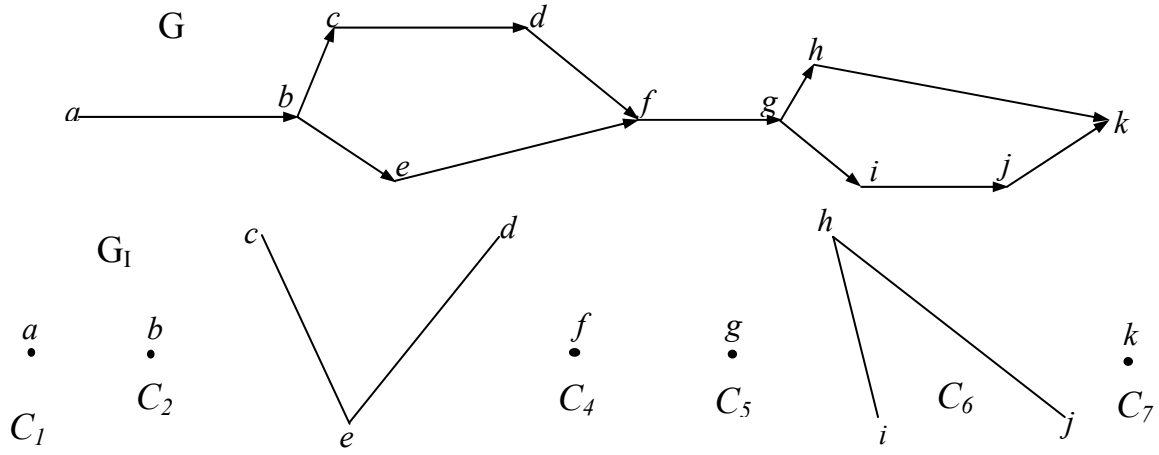


Fig. III-8. Graphe de précédence et composantes connexes du graphe d'indifférence pour l'exemple III-7

Exemple III-7 :

Soient le graphe G et le graphe G_I associé (voir figure III-8).

En ce cas-là, il existe sept composantes connexes, dont deux contiennent plus qu'un symbole. En écrivant les gammes qui respectent G, une conclusion importante s'impose : *les symboles d'une même composante connexe forment des suites "compactes" à l'intérieur des gammes*.

$$\Theta = \begin{cases} ab\underbrace{\mathbf{cde}}_{C_3} fg\underbrace{\mathbf{hij}}_{C_6}k; & ab\underbrace{\mathbf{cde}}_{C_3} fg\underbrace{\mathbf{ihj}}_{C_6}k; & ab\underbrace{\mathbf{cde}}_{C_3} fg\underbrace{\mathbf{ijh}}_{C_6}k; \\ ab\underbrace{\mathbf{ced}}_{C_3} fg\underbrace{\mathbf{hij}}_{C_6}k; & ab\underbrace{\mathbf{ced}}_{C_3} fg\underbrace{\mathbf{ihj}}_{C_6}k; & ab\underbrace{\mathbf{ced}}_{C_3} fg\underbrace{\mathbf{ijh}}_{C_6}k; \\ ab\underbrace{\mathbf{ecd}}_{C_3} fg\underbrace{\mathbf{hij}}_{C_6}k; & ab\underbrace{\mathbf{ecd}}_{C_3} fg\underbrace{\mathbf{ihj}}_{C_6}k; & ab\underbrace{\mathbf{ecd}}_{C_3} fg\underbrace{\mathbf{ijh}}_{C_6}k; \end{cases}$$

Définition III-9 : relation de précédence entre sous-ensembles de symboles "$\lhd$"

Soient deux sous-ensembles disjoints S_1 et S_2 du même ensemble S.

On dit que S_1 est en relation de précédence avec S_2 si tous les symboles de S_1 sont en relation de précédence avec tous les symboles de S_2 :

- soit $(S_1 \lhd S_2) \Leftrightarrow (\forall x_1 \in S_1, \forall x_2 \in S_2 : x_1 < x_2)$;

- soit $(S_2 \lhd S_1) \Leftrightarrow (\forall x_1 \in S_1, \forall x_2 \in S_2 : x_2 < x_1)$.

Lemme L-III.7 :

Soit le graphe d'indifférence G_I tel que $m>1$. Soient C_i et C_j deux composantes connexes quelconques de G_I , $1 \le i < j \le m$, fixées.

Alors, C_i et C_j se trouvent en relation de précédence :

$(C_i \lhd C_j)XOR(C_j \lhd C_i)$.

Démonstration:

Sans diminuer la généralité, nous pouvons supposer que card(C_i)>1 et card(C_j)>1.

Selon la conséquence de la définition d'une composante connexe du graphe d'indifférence, tous deux symboles se trouvant dans des composantes connexes différentes doivent forcément se trouver en relation de précédence, soit dans un sens, soit dans l'autre.

Donc, pour fixer les idées, nous devons montrer que, s'il existe deux symboles a et b, l'un de C_i, l'autre de C_j, tels que a précède b, alors tous les symboles de C_i précèdent tous les symboles de C_j :

$$(\exists a \in C_i, \exists b \in C_j, i \neq j: a<b) \Rightarrow (\forall x \in C_i, \forall y \in C_j : x < y).$$

Nous utilisons le raisonnement par l'absurde. Donc, supposons qu'il y a deux symboles, chacun appartenant à l'une des deux composantes, tels qu'ils ne se précèdent pas dans le sens ci-dessus :

$$\exists x_0 \in C_i,\ \exists y_0 \in C_j : \neg(x_0 < y_0).$$

Il s'ensuit qu'il peut exister deux cas distincts : soit $x_0 ? y_0$, soit $y_0 < x_0$. Nous cherchons à obtenir des contradictions dans tous les deux cas.

1. La contradiction dans le cas $x_0 ? y_0$ est évidente : conformément à la conséquence mentionnée ci-dessus, comme x_0 et y_0 appartiennent aux composantes connexes différentes, ils doivent se trouver en relation de précédence, donc ils ne peuvent pas être indifférents.

2. Si $y_0 < x_0$, nous détaillons l'analyse en faisant quelques observations. Ainsi, une première observation est que *a et y_0 ne peuvent pas être indifférents*, parce qu'ils devraient appartenir à une même composante connexe. Donc, soit $a < y_0$, soit $y_0 < a$. La même observation est valable pour b et x_0. Il en résulte que soit $b < x_0$, soit $x_0 < b$.

Donc, il peut exister l'un des quatre cas suivants :

$$\textbf{2.1}\begin{cases} a < y_0 \\ x_0 < b \end{cases} \qquad \textbf{2.2}\begin{cases} y_0 < a \\ x_0 < b \end{cases} \qquad \textbf{2.3}\begin{cases} a < y_0 \\ b < x_0 \end{cases} \qquad \textbf{2.4}\begin{cases} y_0 < a \\ b < x_0 \end{cases}.$$

*Le cas **2.1***: $a < y_0 < x_0 < b$. Il en résulte que toute gamme $\omega \in \Theta$ s'écrit sous la forme :

$$\omega = ...a.\underbrace{\sigma_1.y_0.\sigma_2}_{\sigma}.x_0.\sigma_3.b...$$

Nous savons que $(a \in C_i) \wedge (x_0 \in C_i)$, donc le résultat du lemme L-III.6 peut s'appliquer à la composante connexe C_i : $\forall s \in \sigma :\ s \in C_i$. Comme $y_0 \in \sigma$, il vient que $y_0 \in C_i$. Mais $y_0 \in C_j$, donc $C_i \cap C_j \neq \varnothing$, ce qui est faux. Par conséquent, nous avons montré que ce cas ne peut pas apparaître. Nous observons que le même raisonnement peut être fait par rapport aux symboles y_0 et b.

*Le cas **2.2***: $y_0 < a < b$ *et* $y_0 < x_0 < b$. L'écriture d'une gamme quelconque ω de Θ dépend de la relation entre a et x_0. Quelle que soit cette relation, il ne peut exister que deux façons distinctes d'écriture de ω :

$$(\omega = ...y_0.\underbrace{\sigma_1.a.\sigma_2.x_0.\sigma_3}_{\sigma}.b...) XOR (\omega = ...y_0.\underbrace{\sigma_1.x_0.\sigma_2.a.\sigma_3}_{\sigma}.b...).$$

Nous raisonnons d'une manière unitaire pour toutes les deux variantes ci-dessus. Du fait que $(b \in C_j) \wedge (y_0 \in C_j)$, en appliquant le résultat du lemme L-III.6 pour la composante connexe C_j, il en résulte que $\forall s \in \sigma :\ s \in C_j$. Comme, par exemple, $x_0 \in \sigma$, il vient que $x_0 \in C_j$. Mais $x_0 \in C_i$, donc $C_i \cap C_j \neq \varnothing$, ce qui est faux. Par conséquent, nous pouvons éliminer ce cas. Nous pourrions traiter de la même façon le cas de a qui se trouve entre y_0 et b.

*Le cas **2.3***: $a < y_0< x_0$ *et* $a < b < x_0$. D'une manière analogue au cas ***2.2***, l'écriture d'une gamme quelconque ω de Θ dépend de la relation entre b et y_0 ; quelle que soit cette relation, il ne peut exister que deux façons distinctes d'écriture de ω :

$$(\omega = ...a.\underbrace{\sigma_1.y_0.\sigma_2.b.\sigma_3}_{\sigma}.x_0...) XOR (\omega = ...a.\underbrace{\sigma_1.b.\sigma_2.y_0.\sigma_3}_{\sigma}.x_0...).$$

Ensuite, nous savons que $(a \in C_i) \wedge (x_0 \in C_i)$ et nous utilisons le résultat du lemme L-III.6 appliqué à la composante connexe C_i: $\forall s \in \sigma :\ s \in C_i$. Comme, par exemple, $b \in \sigma$, il vient que $b \in C_i$. Mais $b \in C_j$, donc $C_i \cap C_j \neq \varnothing$, ce qui est faux. Il résulte que nous devons éliminer ce cas aussi. De plus, observons que nous pouvons considérer aussi le cas de y_0 qui se trouve entre a et x_0.

*Le cas **2.4***: $y_0 < a < b < x_0$. Il en résulte que toute gamme $\omega \in \Theta$ s'écrit sous la forme :

$$\omega = ...y_0.\underbrace{\sigma_1.a.\sigma_2}_{\sigma}.b.\sigma_3.x_0...$$

Nous savons que $(y_0 \in C_j) \wedge (b \in C_j)$. Le résultat du lemme L-III.6 appliqué à la composante connexe C_i conduit à : $\forall s \in \sigma: s \in C_j$. Comme $a \in \sigma$, il vient que $a \in C_j$. Mais $a \in C_i$, donc $C_i \cap C_j \neq \varnothing$, ce qui est faux et nous éliminons ce dernier cas. Le même résultat pourrait être obtenu en raisonnant sur les symboles a et x_0.

Lorsque nous avons éliminé tous les cas possibles à cause d'avoir obtenu des contradictions pour chacun d'eux, la conclusion est que la supposition faite au commencement du raisonnement est fausse.

Le lemme L-III.7 est ainsi prouvé.

Conséquence

Les composantes connexes du graphe d'indifférence sont **totalement ordonnées** :

$$\exists \text{ la permutation } \begin{pmatrix} 1 & 2 & 3 & \dots & m \\ i_1 & i_2 & i_3 & \dots & i_m \end{pmatrix} \text{ telle que : } C_{i_1} \lhd C_{i_2} \lhd C_{i_3} \lhd \dots \lhd C_{i_m},$$

où m est le nombre de composantes connexes.

Exemple III-8 :

Soit l'ensemble de symboles $S=\{a,b,c,d,e,f,g,h,i,j,k\}$.

Soit l'ensemble de gammes d'assemblage $\Omega = \begin{cases} a\ b\ c\ d\ e\ f\ g\ h\ i\ j\ k \\ a\ b\ d\ e\ c\ f\ g\ h\ i\ j\ k \\ a\ b\ d\ c\ e\ f\ h\ g\ i\ j\ k \\ a\ b\ e\ c\ d\ f\ g\ h\ j\ i\ k \end{cases}$, $\Omega \in \mathcal{M}(S)$.

Afin d'illustrer la relation d'ordre total sur l'ensemble des composantes connexes, nous commençons par construire le graphe de précédence et le graphe d'indifférence. L'ensemble de paires de symboles indifférents associé à Ω est :

$P = s(\Omega) = \{(c,d);(d,e);(c,e);(g,h);(i,j)\}$.

Le graphe de précédence G associé à Ω est donné à la figure III-9.

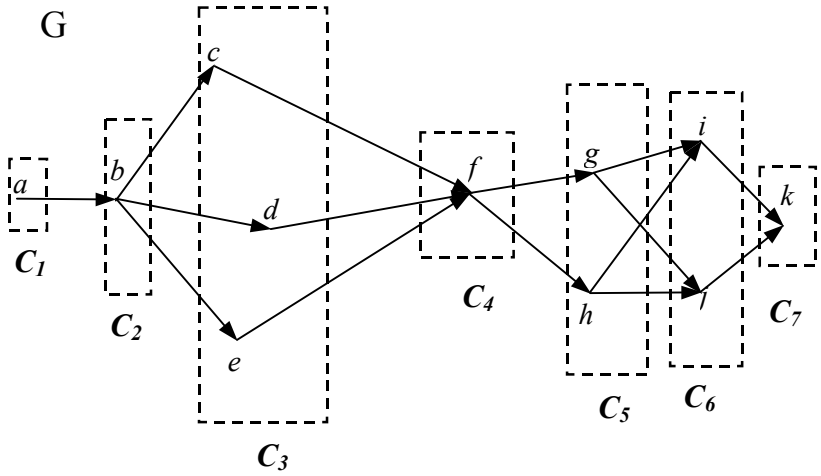


Fig. III-9. Graphe de précédence correspondant à l'ensemble de gammes Ω

Le graphe d'indifférence G_I associé à Ω est montré dans la figure III-10. Il peut être construit directement de l'ensemble de paires de symboles indifférents.

Le graphe G_I a $m=7$ composantes connexes, dont les symboles ont été mis en évidence dans les figures III-9 et III-10 déjà mentionnées. Dans ce cas, nous voyons qu'*aucune composante connexe ne contient de relations de précédence*, donc tous les arcs de précédence restent "à l'extérieur".

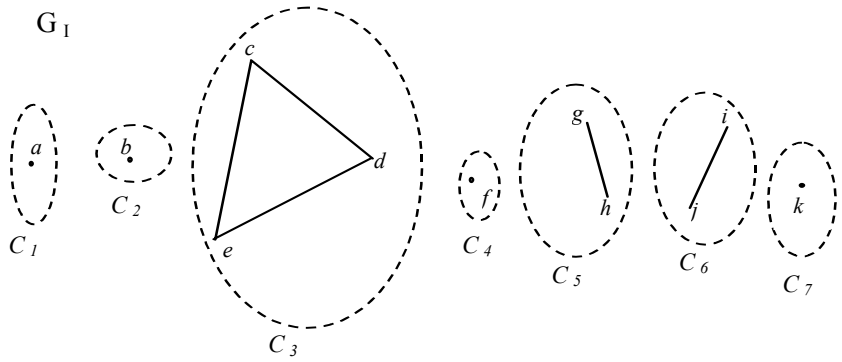


Fig. III-10. Graphe d'indifférence correspondant à l'ensemble de gammes Ω

Il est important de remarquer que *les arcs de précédence restés "à l'extérieur" imposent l'ordre total des composantes connexes*. Nous avons numéroté les composantes selon leur ordre total: $C_1 \lhd C_2 \lhd C_3 \lhd C_4 \lhd C_5 \lhd C_6 \lhd C_7$.

III.5 Théorèmes et résultats principaux

Ce paragraphe est dédié à la présentation des théorèmes, y compris leurs démonstrations basées sur les lemmes et les résultats intermédiaires présentés auparavant. À la fin de cette section nous allons prouver le théorème qui exprime la condition nécessaire et suffisante de représentation équivalente d'un ensemble de gammes par un graphe de précédence.

Définition III-10 : ***segment***

Un segment est une suite formée avec tous les symboles appartenant à *une composante connexe* de G_I.

Observation :

Par exemple, le nombre de segments associés à une composante connexe qui ne contient aucun arc de précédence sera le factoriel du nombre des symboles qu'elle contient (il n'existe aucune contrainte pour arranger les symboles, donc toutes les permutations sont permises).

Notons par $S_i^{(\omega)}$ le segment associé à la composante connexe C_i ($i=1,2...m$) qui apparaît dans la gamme ω. Comme **conséquence du lemme L-III.7** nous formulons le théorème qui définit *la forme d'écriture d'une gamme qui respecte un graphe de précédence*.

THÉORÈME T-III.1 :

Soit un graphe de précédence G et le graphe d'indifférence G_I associé.

Toute gamme qui respecte le graphe G est donnée par une *juxtaposition de segments, telle qui respecte l'ordre total des composantes connexes du graphe* G_I.

Ainsi, si la relation d'ordre total entre les composants connexes de G_I peut être écrite sous la forme $C_1 \lhd C_2 \lhd ... \lhd C_m$, alors : $\forall \omega \in \Theta, \quad \omega = S_1^{(\omega)} S_2^{(\omega)} ... S_m^{(\omega)}$.

Pour illustration, reprenons l'exemple précédent (exemple III-8) :

- il y a *1* segment de C_1: *a*, donc $S_1 \in \{a\}$;
- il y a *1* segment de C_2: *b*; $S_2 \in \{b\}$;
- il y a *6* segments de C_3: *cde*, *ced*, *dec*, *dce*, *ecd*, *edc*; $S_3 \in \{cde, ced, dec, dce, ecd, edc\}$;
- il y a *1* segment de C_4: *f*, donc $S_4 \in \{f\}$;
- il y a *2* segments de C_5: *gh*, *hg*; $S_5 \in \{gh, hg\}$;
- il y a *2* segments de C_6: *ij*, *ji*; $S_6 \in \{ij, ji\}$;
- il y a *1* segment de C_7: *k*; $S_7 \in \{k\}$.

Étant déduite la forme d'écriture des gammes qui respectent G, nous observons que leur ensemble, Θ, *contient* $1 \cdot 1 \cdot 6 \cdot 1 \cdot 2 \cdot 2 \cdot 1 = 24$ *éléments* :

$$\Theta = \begin{cases} a\ b\ cde\ f\ gh\ ij\ k; & a\ b\ ced\ f\ gh\ ij\ k; & a\ b\ dec\ f\ gh\ ij\ k; \\ a\ b\ dce\ f\ gh\ ij\ k; & a\ b\ ecd\ f\ gh\ ij\ k; & a\ b\ edc\ f\ gh\ ij\ k; \\ & & \\ a\ b\ cde\ f\ hg\ ij\ k; & a\ b\ ced\ f\ hg\ ij\ k; & a\ b\ dec\ f\ hg\ ij\ k; \\ a\ b\ dce\ f\ hg\ ij\ k; & a\ b\ ecd\ f\ hg\ ij\ k; & a\ b\ edc\ f\ hg\ ij\ k; \\ & & \\ a\ b\ cde\ f\ gh\ ji\ k; & a\ b\ ced\ f\ gh\ ji\ k; & a\ b\ dec\ f\ gh\ ji\ k; \\ a\ b\ dce\ f\ gh\ ji\ k; & a\ b\ ecd\ f\ gh\ ji\ k; & a\ b\ edc\ f\ gh\ ji\ k; \\ & & \\ a\ b\ cde\ f\ hg\ ji\ k; & a\ b\ ced\ f\ hg\ ji\ k; & a\ b\ dec\ f\ hg\ ji\ k; \\ a\ b\ dce\ f\ hg\ ji\ k; & a\ b\ ecd\ f\ hg\ ji\ k; & a\ b\ edc\ f\ hg\ ji\ k. \end{cases}$$

Pour le développement du raisonnement, il est nécessaire d'introduire ensuite les suivantes **notations** :

$pos: S \times \Omega \to \{1,2,...N\}, pos(a,\omega) = i$
– la fonction qui associe à tout symbol sa position dans une gamme
$(pos(a,\omega) = i \Leftrightarrow \omega(i) = a)$
$\pi: \Theta \times s(\Theta) \to \Theta, \pi(\omega, (a,b)) = \omega', \omega = \sigma_1.ab.\sigma_2, \omega' = \sigma_1.ba.\sigma_2$
– la fonction qui associe à chaque gamme ω de Θ *la gamme symétrique* par rapport à la paire *(a,b)* de symboles indifférents
$p(a,b) \equiv p(b,a)$ – la permutation des symboles *a* et *b*
"." – l'opérateur de juxtaposition

THÉORÈME T-III.2 :

S'il existe deux gammes, ω_1 et ω_2, qui respectent un graphe de précédence connu G, telles que $\omega_1(i) = a$, $\omega_2(j) = a$ et $i<j$, alors il existe une autre gamme ω_3, qui respecte le graphe G, telle que $\omega_3(i+1) = a$ et, de plus, ω_3 peut être obtenue à partir de ω_1 à l'aide d'une suite de permutations.

Démonstration:

L'énoncé du théorème s'écrit :

$$\forall \omega_1, \omega_2 \in \Theta : \omega_1(i) = a, \omega_2(j) = a \ \ et \ \ i < j \ \Rightarrow$$

$$\exists \omega_3 \in \Theta, \exists p \ suite\ de\ permutations : \omega_3(i+1) = a \ \ et \ \ \omega_1 \xrightarrow{p} \omega_3.$$

Si $\omega_1(i) = a$, nous pouvons écrire : $\omega_1 = \sigma_1.a.\sigma_2$, où la suite σ_1 contient *i-1* symboles.

En supposant qu'il n'y a pas des symboles indifférents avec *a* appartenant à σ_2 :

$$\forall\ x \in \sigma_2,\ x<a,$$

ilen résulte que le symbole *a* ne pourra se trouver dans aucune gamme de Θ sur une position supérieure à *i*. Mais il existe la gamme ω_2 où $\omega_2(j) = a$ et $j>i$. Donc, la supposition faite est fausse.

Il s'ensuit que :

$$\exists\ b \in \sigma_2\ le\ premier\ symbole \text{ tel que } b?a.$$

Donc, nous pouvons écrire :

$$\omega_1 = \sigma_1.ax_1x_2...x_qb.\sigma_2' \text{ et } \forall\ x_i,\ i=1...q : a < x_i.$$

Mais $a?b$, donc, conformément au lemme L-III.2 :

$$\forall\ x_i,\ i=1...q,\ (x_i?a) \text{ ou } (x_i?b).$$

Donc, nous avons obtenu que : $\forall x_i,\ i=1...q,\ x_i?b.$

En utilisant **la propriété** Π **de l'ensemble** Θ, nous appliquons la suite de permutations à partir de la gamme ω_1 :

$$\omega_1 \xrightarrow{p(x_p,b)} \sigma_1.ax_1x_2...bx_p.\sigma_2' \in \Theta \xrightarrow{p(x_{p-1},b)} \sigma_1.ax_1x_2...bx_{p-1}x_p.\sigma_2' \in \Theta$$

$$\rightarrow ... \xrightarrow{p(x_1,b)} \sigma_1.abx_1x_2...x_p.\sigma_2' \in \Theta \xrightarrow{p(a,b)} \sigma_1.bax_1x_2...x_p.\sigma_2' = \omega_3 \in \Theta.$$

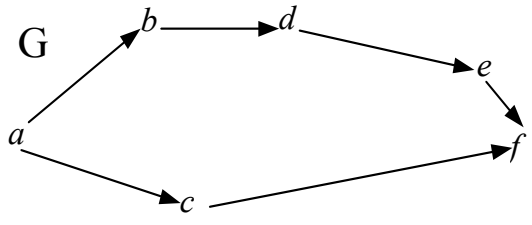
La suite σ_1 contient *i-1* symboles, donc $\omega_3(i+1) = a$, ce que nous avons eu à montrer.

Conséquence

S'il existe 2 gammes, ω_1 et ω_2, qui respectent un graphe de précédence connu G, telles que $\omega_1(i) = a$, $\omega_2(j) = a$ et $i<j$, alors, pour tout nombre naturel *k* situé entre *i* et *j*, il existe une autre gamme ω_3, qui respecte le graphe G, telle que $\omega_3(k) = a$ et, de plus, ω_3 s'obtient par une suite de permutations à partir de ω_1.

Exemple III-9 :

Soit le graphe de précédence G donné dans la figure III-11 ci-dessous :



$$\Leftrightarrow \quad \Theta = \begin{cases} \omega_1 = abcdef \\ \omega_2 = acbdef \\ \omega_3 = abdcef \\ \omega_4 = abdecf \end{cases}$$

Fig. III-11. Graphe de précédence et ensemble équivalent de gammes

Nous appliquons la fonction "s" à l'ensemble Θ afin d'obtenir l'ensemble de paires de symboles indifférents : $s(\Theta) = \{(b,c),(d,c),(e,c)\}$. Soient $i=2$, $j=5$, $i<j$; $\omega_2(i) = c$, $\omega_4(j) = c$. Nous montrons que pour tout $i<k<j$ – c'est à dire, $k=3,4$ – la conséquence ci-dessus est vérifiée :

- pour *k=3*, il existe la gamme ω_1 de Θ telle que $\omega_1(k) = c$; en plus, il existe la suite de permutations (formée par une seule permutation) $p_1=p(c,b)$, qui réalise le transfert $\omega_2 \xrightarrow{p_1} \omega_1$.

- pour *k=4*, il existe la gamme ω_3 de Θ telle que $\omega_3(k) = c$; en plus, il existe la suite de permutations $p_2 = p(c,b).p(c,d)$, qui réalise le transfert $\omega_2 \xrightarrow{p_2} \omega_3$.

Observations :

1) Le raisonnement qui suit est basé sur la forme d'écriture mise en évidence auparavant, qui constitue l'un des plus importants résultats de ce travail.

2) Les exemples fournis auparavant ont montré que la définition III-10 (d'un segment) n'est pas révélatrice au niveau des composantes connexes formées d'un seul symbole. Pratiquement, de **tels symboles ont des positions fixes** à travers l'ensemble des gammes qui respectent un graphe de précédence. Ce qui constitue une bonne raison pour restreindre la notion de "*segment*" aux composantes connexes qui contiennent *plus qu'un symbole*.

Définition III-11 : ***segment au sens restreint***

> Un segment signifie une combinaison possible de symboles appartenant à une composante connexe de G_I, *de cardinal supérieur à 1*.

En ce qui suit, la notion de "*segment*" correspondra à la définition III-11, s'il n'est pas autrement spécifié. Les **notations** faites au sujet des segments changent leurs significations selon la nouvelle définition d'un segment :

m - le nombre des composantes connexes du graphe G_I, *contenant plus qu'un symbole*

S_i - le "*i*"-ème segment, $i = 1,2,...m$

Ainsi, nous reformulons le théorème T-III.1 sous une forme plus claire :

THEOREME T-III.1 reformulé :

> Étant donnés un graphe de précédence G et le graphe d'indifférence G_I associé, les segments associés aux composantes connexes de G_I et l'ensemble de gammes qui respectent G, noté par Θ, nous avons :
>
> $\forall \omega \in \Theta,\ \omega = a\ b\ ...\ c.\boldsymbol{S_1}.d\ e\ ...\ f.\boldsymbol{S_2}.g\ h\ ...\ ...\ i.\boldsymbol{S_m}.j\ k\ ...\ l\ m.$

Exemple III-10 :

> Soit le graphe de précédence de la figure III-12, où nous avons mis en évidence les deux composantes connexes du graphe d'indifférence *ayant plus qu'un symbole* (*m=2*). Donc, il existe *deux segments*.

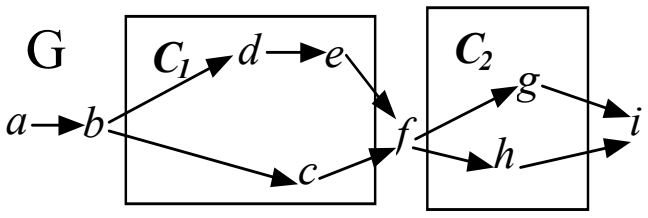


Fig. III-12. Graphe de précédence pour l'exemple III-10

Les positions des symboles *a*, *b*, *f* et *i* restent fixes à travers l'ensemble de gammes qui respectent G :

$\forall \omega \in \Theta,\ \omega = a\ b\ .\boldsymbol{S_1}.\ f\ .\boldsymbol{S_2}.\ i,$ où :

$$\begin{cases} S_1 \in \{cde, dec, dce\} \\ S_2 \in \{gh, hg\} \end{cases};\ \Theta = \begin{cases} a\ b\ \overline{|cde|}\ f\ \overline{|gh|}\ i; & a\ b\ \overline{|dec|}\ f\ \overline{|gh|}\ i; & a\ b\ \overline{|dce|}\ f\ \overline{|gh|}\ i; \\ a\ b\ \overline{|cde|}\ f\ \overline{|hg|}\ i; & a\ b\ \overline{|dec|}\ f\ \overline{|hg|}\ i; & a\ b\ \overline{|dce|}\ f\ \overline{|hg|}\ i. \end{cases}$$

$\text{card}(\Theta) = 1 \cdot 1 \cdot 3 \cdot 1 \cdot 2 \cdot 1 = 6.$

Maintenant nous pouvons formuler une ***conclusion importante*** :

> Le domaine de **la variabilité des gammes** est donné par **la variabilité à l'intérieur des segments** associés aux composantes connexes du graphe d'indifférence.

En utilisant la forme d'écriture d'une gamme quelconque qui respecte un graphe de précédence connu, nous introduisons la **notation** suivante :

$\forall\ \omega \in \Theta$:

$$\omega = \omega(1)\omega(2)...\omega(i_1)\underbrace{\omega(i_1+1)...\omega(i_1+l_1)}_{S_1}\omega(i_1+l_1+1)...\omega(i_2)\underbrace{\omega(i_2+1)...\omega(i_2+l_2)}_{S_2}$$

$$...\omega(i_m)\underbrace{\omega(i_m+1)...\omega(i_m+l_m)}_{S_m}...\omega(N), \qquad l_i > 1, \forall i = 1,2,...m, \quad card(S) = N.$$

THÉORÈME T-III.3 :

Soient deux gammes, ω_1 et ω_2, qui respectent un graphe de précédence connu G – $\omega_1 \in \Theta$, $\omega_2 \in \Theta$ – et qui s'écrivent :

$$\omega_1 = \omega_1(1)...S_1^1...S_2^1 ... S_j^1 ... S_m^1...\omega_1(N);$$

$$\omega_2 = \omega_2(1)...S_1^2...S_2^2 ... S_j^2 ... S_m^2...\omega_2(N).$$

Alors, pour tout $1 \le j \le m$, il existe une suite de permutations qui, à partir de la gamme ω_1, produit la gamme

$$\omega_1' = \omega_1(1)...S_1^1...S_2^1 ... S_j^2 ... S_m^1...\omega_1(N).$$

Démonstration:

Pour simplifier les notations, nous adoptons pour les deux gammes les formes d'écriture suivantes :

$$\omega_1 = ...\underbrace{a_1 a_2 ... a_{l_j}}_{S_j^1}...;\ \omega_2 = ...\underbrace{b_1 b_2 ... b_{l_j}}_{S_j^2}...$$

qui ne mettent en évidence que le segment S_j quelconque fixé.

Les symboles $\{a_{l_k}\}_{k=1...j}$ et $\{b_{l_k}\}_{k=1...j}$ sont les mêmes, mais leur ordres sont différents dans les segments S_j^1 et S_j^2.

À partir de ω_1, en appliquant des permutations, nous allons essayer d'obtenir une gamme qui ait le symbole b_{l_j} sur la position l_j du segment S_j.

- si $a_{l_j} = b_{l_j}$ il n'est besoin d'aucune permutation ;

- sinon, alors $\omega_1(b_{l_j}) < \omega_1(a_{l_j})$, mais $\omega_1(a_{l_j}) = \omega_2(b_{l_j})$, donc $\omega_1(b_{l_j}) < \omega_2(b_{l_j})$ et, conformément à **la conséquence du théorème T-III.2**, il existe une suite de permutations, notée par p_1, telle qui réalise le transfert :

$$\omega_1 \xrightarrow{p_1} \omega_1^1 = ...\underbrace{a_1 a_2 ... a_{l_j-1} b_{l_j}}_{S_j}...$$

D'une manière analogue, nous procédons à partir de ω_1^1 pour obtenir une gamme qui ait le symbole b_{l_j-1} sur la position *lj-1* du segment *Sj*.

- si $\omega_1^1(l_j - 1) = b_{l_j-1}$, ce qui signifie que ce symbole se trouve déjà sur la position *lj-1*, donc il n'est pas nécessaire d'appliquer aucune permutation ;

- sinon, il s'ensuit que $\omega_1^1(b_{l_j-1}) < \omega_2(b_{l_j-1})$; nous utilisons **la conséquence de T-III.2** : il existe une suite de permutations notée par p_2, telle qui produit le transfert :

$$\omega_1^1 \xrightarrow{p_2} \omega_1^2 = ...\underbrace{a_1 a_2 ... a_{l_j-2} b_{l_j-1} b_{l_j}}_{S_j}...$$

Observation :

Pour respecter les conditions du **théorème T-III.2**, il faut avoir la garantie que la suite p_2 de permutations ne change pas la position de b_{l_j} dans la gamme ω_1^1, ce qui signifie que le premier symbole indifférent avec b_{l_j-1} situé à sa droite dans cette gamme ne soit pas b_{l_j}.

Pour montrer la vérité de cette affirmation, nous raisonnons par l'absurde. Supposons, donc, le contraire. En écrivant la gamme ω_1^1 sous la forme :

$$\omega_1^1 = ...\underbrace{b_{l_j-1}\sigma' b_{l_j}}_{S_j}...,$$

nous déduisons que chacun des symboles de la suite σ' doit forcément se trouver en relation de précédence avec b_{l_j-1}. Donc :

$$\forall x \in \sigma' : b_{l_{j-1}} < x .$$

En d'autres mots, *dans toutes les gammes de* Θ, tout symbole de σ' doit se trouver à la droite de b_{l_j-1} et, par conséquent, *entre* b_{l_j-1} *et* b_{l_j}. Mais il existe la gamme ω_2, appartenant à Θ, qui ne remplit pas cette condition. La supposition faite est fausse.

Nous pouvons dire que :

$\exists x \in \sigma'$ *le premier symbole tel que* $x ? b_{l_j-1}$ (donc $x \neq b_{l_j}$).

Ainsi, il est sûr que *la gamme résultée* ω_1^2 *garde le symbole* b_{l_j} *sur la position lj.*

L'algorithme continue pour tout symbole du segment *Sj*. Au dernier pas de l'algorithme, à l'aide d'une suite de permutations notée par p_{l_j-1}, s'obtient la gamme $\omega_1^{l_j-1}$:

$$\omega_1^{l_j-2} \xrightarrow{p_{l_j-1}} \omega_1^{l_j-1} = ...\underbrace{b_1 b_2 ... b_{l_j-1} b_{l_j}}_{S_j}...$$

Nous avons voulu que dans cette gamme le symbole b_2 soit sur la position *2* du segment S_j ; implicitement, le symbole b_1 sera sur la position *1*, d'où il résulte que le segment S_j de $\omega_1^{l_j-1}$ est en fait le segment S_j^2 de ω_2.

Comme conclusion, nous avons obtenu les transferts :

$$\omega_1 \xrightarrow{p_1} \omega_1^1 \xrightarrow{p_2} \omega_1^2 \xrightarrow{p_3} \ldots \xrightarrow{p_{l_j-1}} \omega_1^{l_j-1} = \omega_1' = \ldots \underbrace{b_1 b_2 \ldots b_{l_j}}_{S_j^2} \ldots$$

Donc, il existe une suite de permutations obtenue par la juxtaposition des suites $\{p_k\}_{k=1\ldots l_j-1}$ et notée par $p = p_1.p_2. \ldots p_{l_j-1}$, telle qui réalise le transfert :

$$\omega_1 = \omega_1(1)\ldots S_1^1 \ldots S_2^1 \ldots \boldsymbol{S_j^1} \ldots S_m^1 \ldots \omega_1(N) \xrightarrow{p} \omega_1' = \omega_1(1)\ldots S_1^1 \ldots S_2^1 \ldots \boldsymbol{S_j^2} \ldots S_m^1 \ldots \omega_1(N)$$

ce que nous avons voulu à montrer.

Le résultat obtenu est valable pour tout *j=1...m*, c'est à dire pour tout segment S_j.

Observation :

Les suites de permutations ne changent que l'ordre des symboles de l'intérieur d'un segment fixé ; aussi bien les autres segments, que les symboles qui n'appartiennent à aucun segment restent inchangés.

Exemple III-11 :

Soit le graphe de précédence et le graphe correspondant de la relation d'indifférence (voir figure III-13).

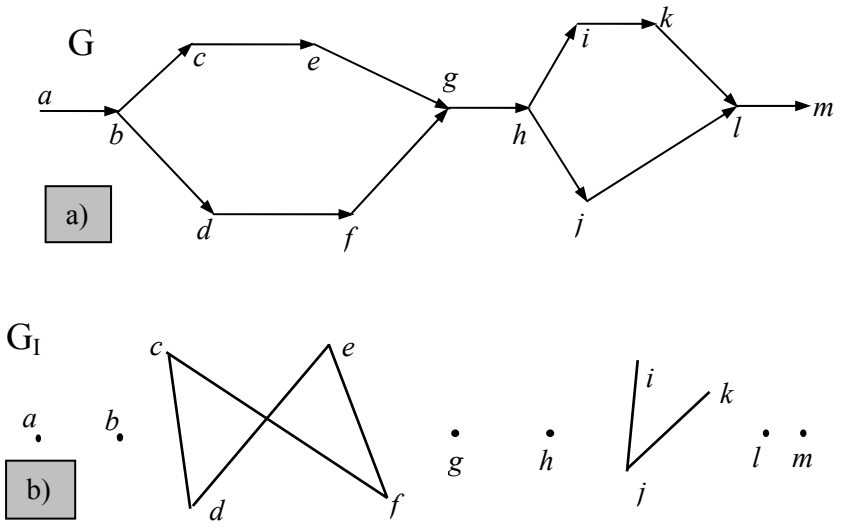


Fig. III-13. a) Graphe de précédence et b) graphe d'indifférence correspondant

G_I a deux composantes connexes dont les cardinaux sont supérieurs à *1*: $C_1=\{c,d,e,f\}$, $C_2=\{i,j,k\}$, donc il existe deux segments $\boldsymbol{S_1}$ et $\boldsymbol{S_2}$ et toute gamme qui respecte le graphe G s'écrit sous la forme :

$\omega = a\ b\ \boldsymbol{S_1}\ g\ h\ \boldsymbol{S_2}\ l\ m.$

Donc, toute paire de gammes, ω_1 et ω_2, appartenant à Θ s'écrit sous la forme :

$\omega_1 = a\ b\ .\ \boldsymbol{S_1^1}\ .\ g\ h\ .\ \boldsymbol{S_2^1}\ .\ l\ m$; $\omega_2 = a\ b\ .\ \boldsymbol{S_1^2}\ .\ g\ h\ .\ \boldsymbol{S_2^2}\ .\ l\ m$.

Soient, par exemple, les gammes :

$$\omega_1 = a\ b\ \underbrace{\boldsymbol{cdfe}}_{S_1^1}\ g\ h\ \underbrace{\boldsymbol{ikj}}_{S_2^1}\ l\ m \quad ; \quad \omega_2 = a\ b\ \underbrace{\boldsymbol{dcef}}_{S_1^2}\ g\ h\ \underbrace{\boldsymbol{ijk}}_{S_2^2}\ l\ m\ .$$

Il existe la suite de permutations $p_1=p(f,e).p(d,c)$, telle qui modifie le segment S_1^1 pour devenir S_1^2 :

$$\omega_1 \xrightarrow{p(f,e)} a\ b\ cd\boldsymbol{ef}\ g\ h\ ikj \xrightarrow{p(d,c)} a\ b\ \boldsymbol{dc}ef\ g\ h\ ikj = \omega_1^1\ ,$$

et il existe la suite de permutations (formée par une seule permutation) $p_2=p(k,j)$, telle qui réalise le transfert du segment S_2^1 vers le segment S_2^2 :

$$\omega_1^1 = a\ b\ dcef\ g\ h\ i\boldsymbol{kj} \xrightarrow{p(k,j)} a\ b\ dcef\ g\ h\ i\boldsymbol{jk} = \omega_2\ .$$

Donc, par la juxtaposition de p_1 et p_2 s'obtient la suite de permutations :

$$p = p_1.p_2 = p(f,e).p(d,c).p(k,j),$$

telle qui réalise le transfert $\omega_1 \xrightarrow{p} \omega_2$.

THÉORÈME T-III.4 : *accessibilité par permutations entre toutes deux gammes qui respectent un graphe de précédence donné*

Soient deux gammes, ω_1 et ω_2, qui respectent un graphe de précédence connu G. Alors il existe une suite de permutations qui, à partir de la gamme ω_1, conduit à la gamme ω_2.

Démonstration:

En tant que gammes qui respectent un graphe de précédence connu, ω_1 et ω_2 peuvent être écrites sous les formes utilisées dans **le théorème T-III.3** :

$$\omega_1 = \omega_1(1)...S_1^1...S_2^1\ ...\ S_j^1\ ...\ S_m^1...\omega_1(N);$$

$$\omega_2 = \omega_2(1)...S_1^2...S_2^2\ ...\ S_j^2\ ...\ S_m^2...\omega_2(N).$$

Nous appliquons **le théorème T-III.3** m fois (pour chaque segment S_j, $j=1,2...m$).

1) Il existe la suite de permutations $p^{(1)}$, telle qui réalise le transfert :

$$\omega_1 \xrightarrow{p^{(1)}} \omega_1^{(1)} = \omega_1(1)...\boldsymbol{S_1^2}...S_2^1\ ...\ S_j^1\ ...\ S_m^1...\omega_1(N)\ .$$

2) Il existe la suite de permutations $p^{(2)}$, telle qui réalise le transfert :

$$\omega_1^{(1)} \xrightarrow{p^{(2)}} \omega_1^{(2)} = \omega_1(1)...S_1^2...\boldsymbol{S_2^2}\ ...\ S_j^1\ ...\ S_m^1...\omega_1(N)\ .$$

. . .

m) Il existe la suite de permutations $p^{(m)}$, telle qui réalise le transfert :

$$\omega_1^{(m-1)} \xrightarrow{p^{(m)}} \omega_1^{(m)} = \omega_1(1)...S_1^2...S_2^2\ ...\ S_j^2\ ...\ \boldsymbol{S_m^2}...\omega_1(N)\ .$$

Nous savons que les symboles qui n'appartiennent pas aux segments gardent leurs positions dans toutes les gammes de Θ :

$\forall\ \omega_1, \omega_2 \in \Theta, \forall\ x \notin S_j, j=1...m,$ si $\omega_1(x)=i,$ alors $\omega_2(x)=i.$

Donc :

$$\omega_1^{(m)} = \omega_1(1)...S_1^2...S_2^2...S_j^2...S_m^2...\omega_1(N) = \omega_2(1)...S_1^2...S_2^2...S_j^2...S_m^2...\omega_2(N) = \omega_2\ .$$

La suite p obtenue par la juxtaposition des suites $p^{(k)}$, $k=1,2...m$: $p=p^{(1)}.p^{(2)}.\ ...\ .p^{(m)}$ réalise le transfert : $\omega_1 \xrightarrow{p} \omega_2$ et le théorème est ainsi montré.

Maintenant nous sommes prêts pour formuler le plus important résultat théorique. Rappelons d'abord quelques **notations** :

Ω - un ensemble donné de gammes d'assemblage
$P = s(\Omega)$ - l'ensemble de paires de symboles indifférents associé à Ω

THÉORÈME T-III.5 : *condition nécessaire et suffisante pour la représentation biunivoque d'un ensemble de gammes par un graphe de précédence*

> Un ensemble donné de gammes est représenté de manière biunivoque par le graphe de précédence associé *si et seulement si cet ensemble possède la propriété* Π .

Démonstration:

Nous savons que pour un ensemble de gammes, *posséder la propriété* Π *c'est posséder cette propriété par rapport à l'ensemble P associé*. Dorénavant, nous allons exprimer cette caractéristique en disant tout simplement que l'ensemble "*possède la propriété* Π".

Nécessité :

Nous savons que $\Omega = \Theta$. Il faut montrer que Ω possède la propriété Π. Ce qui est trivial, parce que nous avons montré que Θ possède la propriété Π (par rapport à soi-même) et que $s(\Omega) = s(\Theta)$.

Suffisance :

Nous savons que Ω possède la propriété Π. Il faut montrer que $\Omega = \Theta$.

Nous allons raisonner par l'absurde. Supposons que $\Omega \neq \Theta$. Comment nous l'avons vu auparavant, $\Omega \subseteq \Theta$, donc la supposition faite conduit à $\Omega \subset \Theta$. Il s'ensuit qu'il existe (au moins) une gamme qui respecte le graphe (donc, qui appartient à Θ) et qui est générée "en plus" (c'est à dire, qui ne se trouve pas dans l'ensemble Ω de départ) :

$$\exists\, \omega \in \Theta \text{ et } \omega \notin \Omega.$$

On applique **le théorème T–III.4** pour la gamme fixée ω :

$$\forall\, \omega' \in \Theta,\ \exists\, p' \text{ une suite de permutations: } \omega \xrightarrow{p'} \omega' .$$

Comme $\Omega \subset \Theta$, alors nous pouvons écrire :

$$\forall\, \omega'' \in \Omega,\ \exists\, p'' \text{ une suite de permutations: } \omega \xrightarrow{p''} \omega'' .$$

Soit $\omega_0 \in \Omega$ une gamme fixée et p_0 la suite de permutations qui réalise la transformation $\omega \xrightarrow{p_0} \omega_0$. Considérons que cette suite de permutations est formée par la juxtaposition des k permutations appliquées par rapport à une suite de paires de symboles indifférents que nous la notons par $\{(a_j,b_j)\}_{j=1,2,\ldots k} \in s(\Omega) = s(\Theta) = P$:

$$p_0 = p(a_1,b_1).p(a_2,b_2). \ldots .p(a_k,b_k) .$$

Ainsi, la transformation réalisée par p_0 peut être détaillé :

$$\omega \xrightarrow{p(a_1,b_1)} \omega_1 \xrightarrow{p(a_2,b_2)} \omega_2 \xrightarrow{p(a_3,b_3)} \ldots \xrightarrow{p(a_{k-1},b_{k-1})} \omega_{k-1} \xrightarrow{p(a_k,b_k)} \omega_0 .$$

Nous allons supposer que, parmi les gammes intermédiaires $\{\omega_j\}_{j=1,2,\ldots k-1}$, il existe des gammes qui ne se trouvent pas dans Ω. Intuitivement, cette situation est représentée dans la figure III-14.

Supposons, par l'absurde, que $\omega_{\kappa-1} \notin \Omega$, alors il en résulte que Ω n'a pas la propriété Π (comme l'opérateur de permutation est réflexif, il y aurait la paire (a_k,b_k) par rapport à laquelle la gamme symétrique de ω_0 ne se trouve pas dans Ω). Nous sommes arrivés à une contradiction, donc $\omega_{\kappa-1} \in \Omega$.

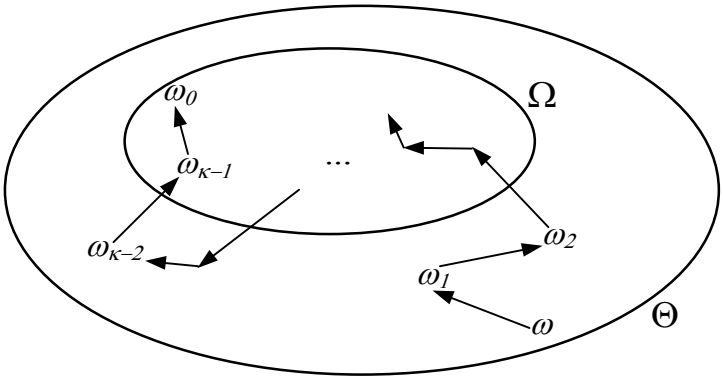


Fig. III-14. Représentation intuitive du "chemin" de permutations entre une gamme ω de Θ et une gamme ω_0 de Ω

En utilisant le même raisonnement nous arrivons à la conclusion que $\omega \in \Omega$, ce qui contredit la supposition faite ($\exists\, \omega \in \Theta$ *et* $\omega \notin \Omega$). Donc cette supposition est fausse. Il résulte que $\Omega = \Theta$ et la suffisance est ainsi montrée.

Ainsi, nous avons fini la démonstration du théorème T-III.5.

Le théorème T-III.5 est **le résultat final** de notre démarche. Il montre dans quelles conditions il existe une correspondance biunivoque entre un ensemble de gammes d'assemblage et un graphe de précédence.

Ensuite, nous allons nous intéresser aux applications des résultats théoriques obtenus. D'abord, il peut être aisément formulé un algorithme pour tester le respect de la propriété Π par un ensemble donné de gammes. Si la réponse est négative, il peut être conçu un algorithme de partage de l'ensemble de gammes en sous-ensembles qui possèdent la propriété Π. Ces sous-ensembles forment une partition. Chaque tel sous-ensemble sera représenté par un seul graphe de précédence. Donc, l'ensemble des graphes de précédence ainsi obtenu représente l'ensemble initial de gammes d'assemblage.

III.6 Exploitation des résultats obtenus

Nous allons présenter **deux algorithmes** qui exploitent les résultats théoriques montrés dans les paragraphes antérieurs, et qui sont des **conséquences directes** du **théorème T-III.5**. Les algorithmes sont bien adaptés à une implémentation en Prolog.

Ces deux algorithmes utilisent les mêmes données d'entrée : un ensemble de gammes, noté par Ω. Le premier algorithme sera appelé l'**algorithme de décision**, lorsqu'il va donner la réponse si l'ensemble donné de gammes possède ou non la propriété Π. Le deuxième algorithme est l'algorithme proprement dit de génération des graphes de précédence à partir de l'ensemble de gammes. Ces deux algorithmes sont basés sur la même idée de base :

choisir une gamme quelconque et *appliquer des suites de permutations pour obtenir successivement des gammes symétriques*.

Mais la liaison entre les deux algorithmes ne consiste pas seulement à utiliser la même idée. Nous allons observer que l'algorithme de décision peut être vu comme une étape initiale du deuxième algorithme. Ce qui signifie que, si le premier donne une réponse négative – l'ensemble de gammes ne possède pas la propriété Π – le deuxième algorithme sera appliqué comme **algorithme de partage** de l'ensemble initial de gammes en sous-ensembles qui ont la propriété Π. Ensuite, à partir de chaque sous-ensemble il sera déduit l'unique graphe de précédence qui le représente, selon les étapes détaillées dans l'exemple III-4. Dans le cas où la réponse de l'algorithme de décision est positive, il n'est pas besoin de partager l'ensemble initial, mais seulement d'appliquer l'algorithme d'obtention du graphe de précédence.

III.6.1 $A_1(\Omega)$ *– l'algorithme de décision*

Tout d'abord, nous devons faire une remarque au sujet du point de départ de cet algorithme. Nous avons dit que la gamme initiale sur laquelle s'applique la première permutation peut être choisie de manière arbitraire. Autrement dit, *le choix de la gamme de départ n'influence pas le résultat de l'algorithme*. La justification de cette affirmation est basée sur le résultat du **théorème T-III.4**, qui établit la propriété d'accessibilité par permutations entre les gammes qui respectent un graphe de précédence.

Plus précisément, nous pouvons détailler cette justification selon les deux réponses possibles données par l'algorithme, comme montré ci-dessous.

- Si l'ensemble Ω possède la propriété Π, alors toute gamme qui lui appartient peut être obtenue par une suite de permutations à partir de toute autre gamme de l'ensemble. Ce qui signifie que, n'importe quelle gamme sera choisie au début, l'algorithme va finir par "couvrir" l'ensemble entier. La fin sera atteinte quand aucune gamme nouvelle ne pourra plus être générée . Donc, le nombre de pas de l'algorithme ne dépend pas de la gamme de départ.
- Si l'ensemble Ω ne possède pas la propriété Π, alors il existe au moins une gamme qui, étant générée par permutation, ne se trouve pas dans l'ensemble Ω. Nous pouvons dire qu'il existe une "rupture" dans la chaîne de permutations, dont la détection nous permet de donner la réponse finale, qui est négative. Cette rupture va forcément être dépistée par l'algorithme, quelle que soit la gamme considérée comme point de départ. Néanmoins*, le nombre de pas* jusqu'à la détection d'une telle rupture *sera variable* en fonction de la gamme de départ.

Une autre observation peut être aussi faite au sujet de la première paire de symboles indifférents qui doivent être permutés à un moment donné. Il s'agit toujours d'un choix arbitraire, qui se justifie par les mêmes raisons que celles listées ci-dessus. Évidemment, dans le premier cas, ce choix n'a aucune influence sur le temps d'exécution, puisque la réponse positive suppose forcément l'application de toutes les permutations. Notons que dans le deuxième cas, étant donnée une gamme, *l'ordre d'application des permutations* va influencer le nombre de pas de l'algorithme.

Ci-après est présentée **la description formelle** de l'algorithme de décision. Nous conservons les notations faites auparavant. Rappelons que la notation $\omega \xrightarrow{p(x,y)} \omega'$ signifie que la gamme ω' s'obtient à partir de la gamme ω par la permutation des symboles indifférents x et y, qui se trouvent en succession directe dans ω. Nous faisons l'extension de cette notation au cas où les deux symboles ne se trouvent pas en succession directe dans ω, en admettant qu'en ce cas le résultat est la gamme vide.

Signification des variables :

continuer – variable logique qui devient "0" lors de l'obtention d'une gamme qui n'appartient pas à l'ensemble de départ (détection d'une "rupture")

M_0 – l'ensemble de gammes nouvelles générées *au cours* du pas *antérieur*

M_1 – l'ensemble de gammes nouvelles générées *au cours* du pas *courant*

M – l'ensemble total de gammes nouvelles générées *jusqu'au* pas *courant*

ω_0 – la gamme initiale, fixée par choix arbitraire

$A_1(\Omega)$

$P \leftarrow s(\Omega)$
$M \leftarrow \{\omega_0\}$
$M_0 \leftarrow \{\omega_0\}$
$M_1 \leftarrow \{\omega_0\}$
continuer ← "1"
Tant que ($M_1 \neq \varnothing$) *et* ($M \neq \Omega$) *et* **continuer**
  $M_1 \leftarrow \varnothing$
  Pour $\omega \in M_0$
    Pour $(x,y) \in P$
      $\omega \xrightarrow{p(x,y)} \omega'$
      Si $\omega' \notin \Omega$
        alors **continuer** ← "0"
        sinon
          Si $\omega' \notin M$
            alors $M_1 \leftarrow M_1 \cup \{\omega'\}$
          Fin Si
      Fin Si
    Répéter
  Répéter
  $M \leftarrow M \cup M_1$
  $M_0 \leftarrow M_1$
Fin Tant que
Si **continuer**
  alors "Ω possède Π"
  autrement "Ω ne possède pas Π"

Exemple III-12 :

Supposons que l'algorithme s'applique à un ensemble de sept gammes :

$$\Omega = \begin{Bmatrix} a\ b\ c\ d\ e\ f\ g \\ a\ c\ b\ d\ e\ f\ g \\ a\ b\ c\ e\ d\ f\ g \\ a\ b\ c\ d\ f\ e\ g \\ a\ c\ b\ e\ f\ d\ g \\ a\ c\ b\ e\ d\ f\ g \\ a\ c\ b\ d\ f\ e\ g \end{Bmatrix} \Rightarrow s(\Omega) = \left\{ \underbrace{(b,c)}_{p_1}; \underbrace{(d,e)}_{p_2}; \underbrace{(d,f)}_{p_3}; \underbrace{(e,f)}_{p_4} \right\}.$$

Dans cet exemple nous voulons intuitivement illustrer l'influence du choix de la gamme initiale sur le nombre de pas de l'algorithme de décision. Malheureusement, il n'existe aucun critère de choix de la gamme initiale, tel qui puisse garantir l'exécution du nombre minimal de pas dans le cas où l'algorithme finit par une réponse négative.

Nous avons montré que l'ensemble de paires de symboles indifférents, $s(\Omega)$, contient quatre éléments. Il peut être facilement vérifié qu'en ce cas l'algorithme de décision va finir par une réponse négative : l'ensemble Ω ne possède pas la propriété Π. Ensuite, nous allons montrer que le temps consommé par l'algorithme dépend d'une combinaison de choix arbitraires : le choix de la gamme initiale, ω_0, et l'ordre des permutations appliquées.

Par exemple, si la gamme ω_4 est choisie comme gamme de départ et si la première permutation est $p(d,f)$, nous constatons que la gamme obtenue n'appartient pas à Ω :

$$\omega_0 = \omega_4 = a\ b\ c\ \boldsymbol{d}\ \boldsymbol{f}\ e\ g \xrightarrow{p(d,f)} \omega' = a\ b\ c\ \boldsymbol{f}\ \boldsymbol{d}\ e\ g \notin \Omega,$$

donc l'algorithme finit en un seul pas. Mais, si nous considérons ω_1 en tant que gamme initiale, alors, quelle que soit la première permutation appliquée, le résultat est une gamme de Ω, ou bien la gamme vide :

$$\omega_0 = \omega_1 = a\ b\ c\ d\ e\ f\ g \begin{cases} \xrightarrow{p(b,c)} a\ \boldsymbol{c}\ \boldsymbol{b}\ d\ e\ f\ g = \omega_2 \in \Omega \\ \xrightarrow{p(d,e)} a\ b\ c\ \boldsymbol{d}\ \boldsymbol{e}\ f\ g = \omega_3 \in \Omega \\ \xrightarrow{p(d,f)} \textit{gamme vide} \\ \xrightarrow{p(e,f)} a\ b\ c\ d\ \boldsymbol{f}\ \boldsymbol{e}\ g = \omega_4 \in \Omega \end{cases}$$

Ensuite, l'algorithme pourrait finir par choisir ω_4 comme gamme suivante à laquelle s'applique la permutation $p(d,f)$. Mais nous ne pouvons faire aucun raisonnement qui puisse nous conduire nécessairement vers ce choix.

L'algorithme de décision n'est pas polynomial, ce qui représente un inconvénient. Cette conclusion, qui nous est suggérée par les observations susmentionnées, peut être dégagée comme suite de l'estimation de la complexité. Mais elle peut être déduite d'une autre façon, si nous observons la complexité combinatoire intrinsèque du problème de décision lui-même. Nous avons montré l'inexistence des critères de réduction de la combinatoire.

III.6.2 $A_2(\Omega)$ *– l'algorithme de partage*

L'algorithme de partage s'applique comme *étape suivante* de l'algorithme de décision quand ce dernier finit par *une réponse négative* : l'ensemble Ω de gammes ne peut pas être représenté par un seul graphe de précédence. Il en résulte qu'il existe plusieurs graphes de précédence qui lui correspondent, chacun d'eux étant le correspondant biunivoque d'un sous-ensemble de gammes.

Lorsque nous voulons trouver ces graphes de précédence, il faut donc chercher *les sous-ensembles qui aient la propriété* Π, évidemment, *par rapport* à des sous-ensembles de paires de symboles indifférents (*sous-ensembles de* $P = s(\Omega)$). Cette idée conduit à la nécessité de construire **une partition** de Ω.

Une question se pose : est-ce qu'il existe une seule partition ? La réponse est négative. Pour réduire l'espace des solutions possibles, remarquons que notre objectif peut se formuler en termes de problème d'optimisation :

trouver **l'ensemble minimal de graphes de précédence** qui représente l'ensemble donné de gammes.

Une fois que ce problème est résolu, il nous reste encore le problème de l'utilisation du résultat. Parfois, dans la conception des systèmes d'assemblage, il est convenable de conserver seulement *le graphe de précédence maximal* en tant que modèle du processus d'assemblage, en éliminant les autres. La raison réside dans le but d'obtenir une meilleure flexibilité du système d'assemblage, puisque le graphe maximal introduit le nombre minimal de contraintes de précédence sur l'ensemble des tâches d'assemblage (voir aussi [CHEN 96]).

Nous nous intéressons à ce qui devient notre problème formulé ci-dessus. Il est évident que trouver le nombre minimal de graphes de précédence à partir d'un ensemble de gammes est équivalent à :

> trouver de manière algorithmique **la partition minimale** des sous-ensembles qui aient la propriété Π.

Intuitivement, cela signifie qu'au premier pas nous allons chercher **le sous-ensemble maximal** qui ait cette propriété. Cette idée de base se propage ensuite aux gammes restées après avoir détaché ce sous-ensemble de l'ensemble total. L'algorithme finit quand l'ensemble des gammes restées devient vide. Nous allons concevoir l'algorithme basé sur ce principe, puis nous allons montrer que son résultat est équivalent à l'ensemble minimal de graphes de précédence.

L'algorithme de partage utilise le même principe que l'algorithme de décision : *la génération des gammes symétriques par permutation des symboles indifférents*. Nous ne connaissons pas a priori l'ensemble de paires de symboles indifférents par rapport auquel le sous-ensemble maximal pourrait posséder la propriété Π. C'est ainsi que le problème principal qui se pose est d'établir **l'ensemble de paires de symboles indifférents** par rapport auquel nous cherchons le remplissage de la propriété Π par un sous-ensemble de gammes.

Évidemment, dans une itération quelconque de l'algorithme, cet ensemble de paires est un sous-ensemble de $P = s(\Omega')$, où Ω' est une notation générique pour l'ensemble de gammes restées à cette itération. Premièrement, nous vérifions le remplissage de la propriété désirée par l'ensemble Ω'. Pratiquement, cela signifie l'application de *l'algorithme de décision* décrit auparavant. Si nous obtenons une réponse négative, il en résulte que nous devons vérifier ensuite s'il existe des sous-ensembles avec cette propriété par rapport aux sous-ensembles de P contenant, dans un premier temps, moins d'un élément. En d'autres mots, si $\text{card}(P) = k$, nous construisons les parties de P qui aient $k-1$ éléments. Pour chacune de ces parties, nous cherchons les sous-ensembles qui pourraient avoir la propriété Π. S'il y en a, nous choisissons le(s) sous-ensemble(s) maximal(aux). Sinon, nous reprenons la recherche par rapport aux parties formées de $k-2$ éléments, etc.

De ce que nous avons décrit ci-dessus il résulte que l'algorithme de décision peut être vu comme cas particulier de l'algorithme de partage. Une autre observation doit être faite au sujet des parties de $s(\Omega')$. Il n'existe aucun critère de choix qui puisse nous montrer dès le début la partie qui conduit au sous-ensemble cherché. Il s'ensuit que nous devons considérer *toutes les parties* d'un ensemble de paires associé à un ensemble Ω' de gammes restées à un moment donné, en ordre décroissant de leurs cardinaux. Donc, la complexité du problème est principalement due à la nécessité de construire l'ensemble de parties d'un ensemble, qui conduit à une combinatoire de type factoriel.

Les observations concernant la manière d'application des permutations – le choix de la gamme initiale à laquelle s'applique la première permutation et le choix, toujours arbitraire, de la paire des symboles indifférents qui seront permutés à un moment donné dans une gamme fixée – restent valables dans le cas de l'algorithme de partage, comme pour l'algorithme de décision.

Néanmoins, nous pouvons faire quelques *observations*, une fois que nous avons fixé l'ensemble de paires de symboles indifférents par rapport auquel nous cherchons le remplissage de la propriété Π. Notons cet ensemble par *R*.

Le choix d'une première gamme – comme point de départ pour l'application des permutations – peut être restreint aux gammes qui contiennent au moins *une succession directe* des symboles indifférents se trouvant parmi les paires de *R*. Supposons, par exemple, qu'une telle gamme soit notée par ω_0. Remarquons que, s'il existe un sous-ensemble avec la propriété Π par rapport à l'ensemble *R*, il doit nécessairement contenir la gamme ω_0 (comme suite des lemmes L-III.3 et L-III.5). Donc, si la génération des symétriques à partir de ω_0 nous permet de conclure qu'un tel sous-ensemble n'existe pas, alors la même conclusion reste valable quelle que soit la gamme considérée. *Il n'est donc pas nécessaire de reprendre la génération des gammes symétriques à partir de chaque gamme de l'ensemble testé à une itération donnée*.

De plus, nous pouvons faire une *remarque* au sujet du nombre des solutions : en général, il peut arriver que le résultat de la recherche ne soit pas unique. Ce qui est une conséquence du fait qu'il peut exister *plusieurs sous-ensembles maximaux* avec la propriété Π, du même cardinal, mais qui contiennent des gammes différentes. Nous montrons une telle situation dans l'exemple suivant.

Exemple III-13 :

Soient l'ensemble de gammes Ω et l'ensemble de paires de symboles indifférents associé :

$$\Omega = \begin{Bmatrix} a\ b\ c\ d\ e\ f\ g\ h\ i & a\ c\ d\ b\ e\ f\ g\ h\ i \\ a\ b\ d\ c\ e\ f\ h\ g\ i & a\ d\ c\ b\ e\ f\ h\ g\ i \\ a\ b\ c\ d\ e\ f\ h\ g\ i & a\ c\ d\ b\ e\ f\ h\ g\ i \\ a\ b\ d\ c\ e\ f\ g\ h\ i & a\ d\ c\ b\ e\ f\ g\ h\ i \end{Bmatrix} \Rightarrow \mathbb{s}(\Omega) = \{(b,c),(c,d),(b,d),(g,h)\}$$

Tout d'abord, remarquons que Ω n'a pas la propriété Π. Par exemple, nous pouvons mettre en évidence deux sous-ensembles de Ω qui ont cette propriété par rapport au même sous-ensemble de paires de $\mathbb{s}(\Omega)$, noté par $R = \{(c,d),(g,h)\}$:

$$\Omega \supset \Omega_1 = \begin{Bmatrix} a\ b\ \mathbf{c}\ \mathbf{d}\ e\ f\ \mathbf{g}\ \mathbf{h}\ i \\ a\ b\ \mathbf{d}\ \mathbf{c}\ e\ f\ \mathbf{h}\ \mathbf{g}\ i \\ a\ b\ \mathbf{c}\ \mathbf{d}\ e\ f\ \mathbf{h}\ \mathbf{g}\ i \\ a\ b\ \mathbf{d}\ \mathbf{c}\ e\ f\ \mathbf{g}\ \mathbf{h}\ i \end{Bmatrix} \qquad \Omega \supset \Omega_2 = \begin{Bmatrix} a\ \mathbf{c}\ \mathbf{d}\ b\ e\ f\ \mathbf{g}\ \mathbf{h}\ i \\ a\ \mathbf{d}\ \mathbf{c}\ b\ e\ f\ \mathbf{h}\ \mathbf{g}\ i \\ a\ \mathbf{c}\ \mathbf{d}\ b\ e\ f\ \mathbf{h}\ \mathbf{g}\ i \\ a\ \mathbf{d}\ \mathbf{c}\ b\ e\ f\ \mathbf{g}\ \mathbf{h}\ i \end{Bmatrix}$$

Si nous considérons *R* comme étant fixé, nous observons que les deux sous-ensembles ci-dessus sont des sous-ensembles maximaux qui ont la propriété Π par rapport à *R*. Ils ont le même nombre de gammes (quatre), mais ils contiennent des gammes différentes. Donc, tous les deux peuvent constituer des solutions de la recherche concernée à un moment donné par l'algorithme de partage. L'obtention de l'un ou de l'autre dépend du choix de la gamme initiale à laquelle s'applique la première permutation.

Le dernier exemple facilite la formulation de la

Proposition P-III.1 :

Soient deux ensembles de gammes, Ω_1 et Ω_2, formés de symboles d'un même ensemble *S*. Si $\mathbb{s}(\Omega_1)=\mathbb{s}(\Omega_2)$ et si les deux ensembles possèdent la propriété Π, alors $\text{card}(\Omega_1)=\text{card}(\Omega_2)$.

Justification :

La propriété Π étant remplie par les deux ensembles, il s'ensuit qu'ils peuvent être respectivement représentés de manière biunivoque par deux graphes de précédence G_1 et G_2. Lorsque $s(\Omega_1)=s(\Omega_2)$, les graphes d'indifférence respectivement associés aux G_1 et G_2 ont *les mêmes composantes connexes contenant plus qu'un élément*. Conformément au **théorème T-III.1** – concernant la forme d'écriture des gammes qui respectent un graphe de précédence – le nombre de ces gammes est dicté exclusivement par le nombre et la structure des *segments*. Rappelons que les segments peuvent être vus comme correspondant de façon biunivoque aux composantes connexes qui ont des cardinaux supérieurs à *1*.

Donc, les gammes de Ω_1 et de Ω_2 contiennent *les mêmes segments*. Ce qui conduit à $card(\Omega_1)=card(\Omega_2)$.

Nous avons vu qu'en général la recherche de la partition minimale peut conduire à plusieurs solutions. Évidemment, il ne s'agit pas du nombre des sous-ensembles résultants, mais de leurs compositions. Il en résulte qu'il peut exister plusieurs ensembles minimaux de graphes de précédence qui représentent un ensemble donné de gammes. Nous nous contentons que l'algorithme proposé fournisse *une* solution du problème de partage : *la première* trouvée. Sa **description formelle** est présentée ci-après.

Pour assurer la lisibilité et la compréhension de l'algorithme principal, nous allons utiliser deux procédures.

La procédure **parties**(*R*, *k*) reçoit comme données d'entrée un ensemble *R* et un nombre naturel *k* strictement positif. Elle a comme résultat l'ensemble *indexé* de parties de *R* qui contiennent *k* éléments. À une seule exception, aux moments d'appel de cette procédure *R* est un ensemble de paires non ordonnées de symboles

La procédure **contient_sous_ens_avec_pi**(X, *R*) reçoit comme données d'entrée un ensemble X de gammes et un ensemble *R* de paires de symboles, ayant comme résultat *un* sous-ensemble maximal de X qui a la propriété Π par rapport à l'ensemble *R*. Notons que, au moment d'appel de cette procédure au sein de l'algorithme principal, on a toujours $R \subseteq s(X)$. Le résultat dépend de la gamme choisie comme point de départ pour l'application des permutations. Notons aussi que cette procédure fournit le résultat "∅" si X ne contient aucun sous-ensemble avec la propriété susmentionnée. Nous considérons que cette procédure doit être détaillée elle aussi, comme étant le point essentiel de l'algorithme de partage. Elle représente de fait une extension de l'algorithme de décision.

Signification des notations :

PARTITION = $\{\Omega_1, \Omega_2, \ldots \Omega_q\}$ la partition minimale de Ω

PAIRES_PARTITION = $\{R_1, R_2, \ldots R_q\}$, où le sous-ensemble Ω_i possède la propriété Π par rapport à l'ensemble R_i de paires de symboles indifférents

$\underline{\mathbb{A}_2(\Omega)}$

```
X ← Ω
Si (card(X)=1) alors
      PARTITION ← X
      PAIRES_PARTITION ← ∅
   sinon
      PARTITION ← ∅
      PAIRES_PARTITION ← ∅
```

```
        Tant que (X≠∅)
            P ← s(X)
            k ← card(P)
            cardmax ← 0

            Tant que (k>0) et (cardmax=0)
                R ← parties(P, k)
                i ← 1
                Tant que (i<=card(R))
                    Ψ ← contient_sous_ens_avec_pi(X, Ri)
                    Si (card (Ψ)>cardmax) alors
                        cardmax ← card(Ψ)
                        Ψmax ← Ψ
                        Rmax ← Ri
                    Fin Si
                    i ← i+1
                Fin Tant que
                Si (cardmax>0) alors
                        PARTITION ← PARTITION ∪ Ψmax
                        PAIRES_PARTITION←PAIRES_PARTITION ∪ Rmax
                    sinon k ← k-1
                Fin Si
            Fin Tant que

            Si (k>0)
                alors X ← X - Ψmax
                sinon
                    PARTITION_1 ← parties(X, 1)
                    PARTITION ←PARTITION ∪ PARTITION_1
                    Pour i de 1 à card(PARTITION_1)
                        PAIRES_PARTITION ← PAIRES_PARTITION ∪ {∅}
                    Répéter
                    X ← ∅
            Fin Si

        Fin Tant que
    Fin Si
```

Procédure **contient_sous_ens_avec_pi**(X, *R*)

Signification des notations :

continuer – variable logique qui devient "0" lors de l'obtention d'une gamme qui "excède" l'ensemble X de gammes

Ψ_0 – l'ensemble de gammes nouvelles générées *au cours* du pas *antérieur*

Ψ_1 – l'ensemble de gammes nouvelles générées *au cours* du pas *courant*

Ψ – l'ensemble total de gammes nouvelles générées *jusqu'au* pas *courant*

#1. Choisir ω_0 telle que : $(\omega_0 \in X) \wedge (\omega_0 = \sigma_1.xy.\sigma_2) \wedge ((x,y) \in R)$.

$\Psi \leftarrow \{\omega_0\}$

$\Psi_0 \leftarrow \{\omega_0\}$

$\Psi_1 \leftarrow \{\omega_0\}$
continuer ← "1"

Tant que ($\Psi \neq X$) *et* ($\Psi_1 \neq \varnothing$) *et* (**continuer**)
  $\Psi_1 \leftarrow \varnothing$

  Pour $\omega \in \Psi_0$
    Pour $p \in R$
      $\omega \xrightarrow{p} \omega_s$
      Si ($\omega_s \notin X$)
        alors **continuer** ← "0"
        sinon
          Si ($\omega_s \notin \Psi$)
            alors $\Psi_1 \leftarrow \Psi_1 \cup \{\omega_s\}$
          Fin Si
      Fin Si
    Répéter
  Répéter

  $\Psi \leftarrow \Psi \cup \Psi_1$
  $\Psi_0 \leftarrow \Psi_1$
Fin Tant que

Si NOT(**continuer**)
  alors $\Psi \leftarrow \varnothing$
Fin Si

Retourner{Ψ}

Observation :

Nous avons montré que pratiquement, à tout appel de la procédure ci-dessus au cours de l'algorithme principal, il ne s'agit pas d'un ensemble quelconque R de paires de symboles, mais d'un sous-ensemble de $\mathfrak{s}(X)$: $R \subseteq \mathfrak{s}(X)$. Il en résulte que le choix de la gamme ω_0 de la manière décrite au point noté par #1 aura toujours un résultat consistant, puisque l'existence d'une telle gamme dans X est garantie par **le lemme L-III.5**.

En ce qui suit, nous présentons un exemple de déroulement détaillé de l'algorithme de partage (voir également [HENR 99]).

Exemple III-14 :

Nous appliquons l'algorithme de partage à l'ensemble de gammes :

$$\Omega = \begin{cases} a\ b\ c\ d\ e\ f\ g\ h & a\ c\ b\ d\ g\ f\ e\ h \\ a\ c\ b\ d\ g\ e\ f\ h & a\ c\ b\ d\ e\ g\ f\ h \\ a\ b\ c\ d\ g\ e\ f\ h & a\ b\ c\ d\ g\ f\ e\ h \end{cases}$$

Pas 1: $X = \Omega$; $P = \mathfrak{s}(X) = \{(b,c), (e,f), (f,g), (e,g)\}$;
$R =$ **parties**$(P, 4) = \{P\}$;
X ne possède pas la propriété Π par rapport au seul élément de R ;

Pas 2: R = **parties**(*P*, *3*); card(R) = *4* ;
X ne contient aucun sous-ensemble avec la propriété Π par rapport à aucun sous-ensemble R_i de R, *i=1,2,3,4* ;

Pas 3: R = **parties**(*P*, *2*); card(R) = *6* ;
X contient deux sous-ensembles ayant la propriété Π par rapport aux éléments de R :

$$\Psi_1 = \begin{cases} a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{e}\ \boldsymbol{f}\ h \\ a\ \boldsymbol{c}\ \boldsymbol{b}\ d\ g\ \boldsymbol{f}\ \boldsymbol{e}\ h \\ a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{f}\ \boldsymbol{e}\ h \\ a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{e}\ \boldsymbol{f}\ h \end{cases} ; R_1 = \{(b,c), (e,f)\} ;$$

$$\Psi_2 = \begin{cases} a\ c\ b\ d\ g\ \boldsymbol{f}\ \boldsymbol{e}\ h \\ a\ c\ b\ d\ \boldsymbol{g}\ \boldsymbol{e}\ \boldsymbol{f}\ h \\ a\ c\ b\ d\ \boldsymbol{e}\ \boldsymbol{g}\ f\ h \end{cases} ; R_2 = \{(e,f), (e,g)\} ;$$

cardmax = card(Ψ_1) = *4*, donc Ψmax = Ψ_1 et Rmax = R_1 ;
PARTITION = $\{\Psi_1\}$; PAIRES_PARTITION = $\{R_1\}$;
En éliminant Ψ_1 de X, il résulte $X = \begin{cases} a\ b\ c\ d\ e\ f\ g\ h \\ a\ c\ b\ d\ e\ g\ f\ h \end{cases}$;

Pas 4: $P = s(X) = \{(b,c), (g,f)\}$;
R = **parties**(*P*, *2*) = $\{P\}$;
X n'a pas la propriété Π par rapport au seul élément de l'ensemble R ;

Pas 5: R = **parties**(*P*, *1*); card(R) = *2* ;
X ne contient aucun sous-ensemble avec la propriété Π par rapport à aucun sous-ensemble R_i de R, *i=1,2* ; il résulte cardmax=*0*, donc :
PARTITION = $\{\Psi_1\} \cup$ **parties**(X, *1*) = $\{\Psi_1, \{a\ b\ c\ d\ e\ f\ g\ h\}, \{a\ c\ b\ d\ e\ g\ f\ h\}\}$;
PAIRES_PARTITION = $\{R_1, \varnothing, \varnothing\}$;
X devient $\varnothing$;

Pas 6: PARTITION = $\{\Psi_1, \Psi_2, \Psi_3\}$;
PAIRES_PARTITION = $\{R_1, R_2, R_3\}$, où :

$$\Psi_1 = \begin{cases} a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{e}\ \boldsymbol{f}\ h \\ a\ \boldsymbol{c}\ \boldsymbol{b}\ d\ g\ \boldsymbol{f}\ \boldsymbol{e}\ h \\ a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{f}\ \boldsymbol{e}\ h \\ a\ \boldsymbol{b}\ \boldsymbol{c}\ d\ g\ \boldsymbol{e}\ \boldsymbol{f}\ h \end{cases} ; R_1 = \{(b,c), (e,f)\} ;$$

$\Psi_2 = \{a\ b\ c\ d\ e\ f\ g\ h\}$; $R_2 = \varnothing$;
$\Psi_3 = \{a\ c\ b\ d\ e\ g\ f\ h\}$; $R_3 = \varnothing$.

Le résultat listé au dernier pas représente la partition trouvée par l'algorithme de partage ayant la description présentée ci-dessus. Rappelons que l'algorithme est basé sur l'idée de la recherche récursive **des sous-ensembles maximaux qui ont la propriété** Π. En ce point-là nous montrons que cette solution est en fait **une partition minimale** de l'ensemble donné.

Proposition P-III.2 :

Soit un ensemble de gammes Ω. La partition $\{\Omega_i\}_{i=1,2,...p}$ obtenue en appliquant l'algorithme de partage A_2 à l'ensemble Ω est minimale.

Démonstration :

La partition $\{\Omega_i\}_{i=1,2,...p}$ contient des sous-ensembles qui ont la propriété Π et les ensembles de paires de symboles indifférents respectivement associés, $\{s(\Omega_i)\}_{i=1,2,...p}$, sont des sous-ensembles de $s(\Omega)$, sans constituer forcément une partition de ce dernier.

Nous raisonnons par l'absurde, en supposant qu'il existe une autre partition $\{\Omega_j\}_{j=1,2,...q}$ de Ω, telle que $p>q$. Notons par Ω_s le plus grand sous-ensemble de la première partition et par Ω_t le plus grand des sous-ensembles de la deuxième partition :

$$\Omega \supseteq \Omega_s : \text{card}(\Omega_s) = \max_{i=1,p}\{\text{card}(\Omega_i)\};$$

$$\Omega \supseteq \Omega_t : \text{card}(\Omega_t) = \max_{j=1,q}\{\text{card}(\Omega_j)\}.$$

Comme $p>q$, nous obtenons la relation évidente :

$\text{card}(\Omega_t)>\text{card}(\Omega_s)$.

Notons $k=\text{card}(s(\Omega_s))$, où, évidemment, $s(\Omega_s)$ est une partie de $s(\Omega)$. Étant donnée la description de l'algorithme, le sous-ensemble Ω_s a été obtenu comme *le plus grand* des sous-ensembles qui ont la propriété Π par rapport aux parties de $s(\Omega)$ avec k éléments.

Notons $l=\text{card}(s(\Omega_t))$, $s(\Omega_t)\subseteq s(\Omega)$. Remarquons que Ω_t est lui aussi un sous-ensemble ayant la propriété Π par rapport à une partie de $s(\Omega)$ avec l éléments. Donc, Ω_t a été lui aussi généré à une certaine itération de l'algorithme, mais il n'a pas été choisi comme étant le maximal.

Nous pouvons avoir les situations suivantes :

- $k \neq l$

Les deux sous-ensembles, Ω_s et Ω_t, ont été générés au cours d'itérations distinctes. La seule raison pour laquelle Ω_t n'a pas été choisi comme maximal pourrait être la détection au cours de la même itération d'un sous-ensemble Ω_t', encore plus grand, qui possède la propriété Π. Ce sous-ensemble, puisque résultant de l'algorithme, doit forcément fait partie de la partition $\{\Omega_i\}_{i=1,2,...p}$. Mais, comme $\text{card}(\Omega_t')>\text{card}(\Omega_t)>\text{card}(\Omega_s)$, il résulte : $\text{card}(\Omega_t)>\text{card}(\Omega_s)$. Ce qui ***contredit*** le fait que Ω_s est le plus nombreux des sous-ensembles de la partition $\{\Omega_i\}_{i=1,2,...p}$.

- $k=l$

Les deux sous-ensembles, Ω_s et Ω_t, ont été générés au cours de la même itération. Dans ce cas il peut apparaître deux sous-cas :

  - $s(\Omega_s) \neq s(\Omega_t)$

  Les ensembles de paires de symboles indifférents par rapport auxquels Ω_s et Ω_t possèdent respectivement la propriété Π sont distincts. Comme $\text{card}(\Omega_t)>\text{card}(\Omega_s)$, il vient que l'algorithme aurait du choisir l'ensemble Ω_t en tant que maximal, au lieu de Ω_s. Ce qui est une ***contradiction***.

  - $s(\Omega_s) = s(\Omega_t)$

  Ω_s et Ω_t ont la propriété Π par rapport au même ensemble de paires de symboles indifférents. Mais, conformément à **la proposition P-III.1**, les deux sous-ensembles devraient avoir le même cardinal. Nous sommes arrivés à une ***contradiction***.

Lorsque nous avons obtenu des contradictions pour tous les cas possibles, il s'ensuit que la supposition faite est fausse. Donc, la partition résultant de l'algorithme A_2 est minimale. La démonstration est finie.

Exemple III-15 :

Nous reprenons l'exemple III-14. Nous avons vu qu'à partir de l'ensemble de gammes :

$$\Omega = \begin{cases} a\,b\,c\,d\,e\,f\,g\,h & a\,c\,b\,d\,g\,f\,e\,h \\ a\,c\,b\,d\,g\,e\,f\,h & a\,c\,b\,d\,e\,g\,f\,h \\ a\,b\,c\,d\,g\,e\,f\,h & a\,b\,c\,d\,g\,f\,e\,h \end{cases},$$

la partition en sous-ensembles avec la propriété Π, telle qu'issue de l'algorithme de partage, est $\{\Psi_i\}_{i=1,2,3}$, où :

$$\Psi_1 = \begin{cases} a\,\boldsymbol{b}\,\boldsymbol{c}\,d\,g\,\boldsymbol{e}\,\boldsymbol{f}\,h \\ a\,\boldsymbol{c}\,\boldsymbol{b}\,d\,g\,\boldsymbol{f}\,\boldsymbol{e}\,h \\ a\,\boldsymbol{b}\,\boldsymbol{c}\,d\,g\,\boldsymbol{f}\,\boldsymbol{e}\,h \\ a\,\boldsymbol{b}\,\boldsymbol{c}\,d\,g\,\boldsymbol{e}\,\boldsymbol{f}\,h \end{cases} ; \; s(\Psi_1) = \{(b,c), (e,f)\} ;$$

$\Psi_2 = \{a\,b\,c\,d\,e\,f\,g\,h\}$; $s(\Psi_2) = \varnothing$;

$\Psi_3 = \{a\,c\,b\,d\,e\,g\,f\,h\}$; $s(\Psi_3) = \varnothing$.

Nous utilisons cet exemple pour arriver au but final de notre démarche : l'obtention de l'ensemble minimal de graphes de précédence *équivalent* à un ensemble donné de gammes. Chaque sous-ensemble de la partition ci-dessus possède la propriété Π. Donc, chacun d'eux est représenté par un seul graphe de précédence, comme montré à la figure III-15.

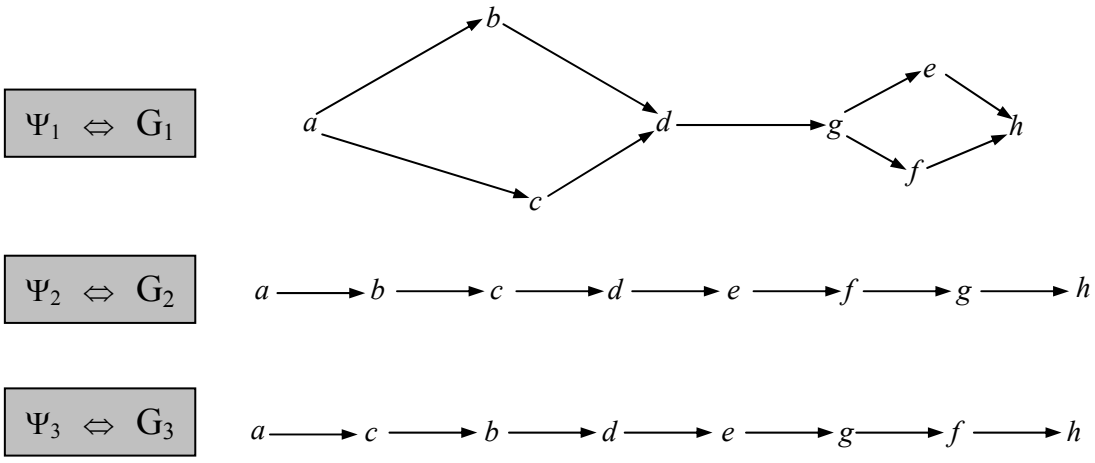


Fig. III-15. Ensemble minimal de graphes de précédence équivalent à l'ensemble Ω de gammes

Nous avons montré que **la partition** $\{\Psi_i\}_{i=1,2,3}$ est **minimale**. Ce qui nous permet d'affirmer ensuite que **l'ensemble de graphes de précédence** $\{G_i\}_{i=1,2,3}$ que nous avons déduit est **minimal**.

Le problème de la génération des graphes de précédence à partir d'un ensemble de gammes d'assemblage est ainsi résolu. Ce qui s'écrit en ce cas :

$$\boxed{\Omega \Leftrightarrow \{G_i\}_{i=1,2,3}}$$

III.7 Conclusion

Le but de ce chapitre a été de présenter **une approche systématique** de "*conversion*" *d'un ensemble de gammes d'assemblage en graphes de précédence*. Nous pouvons dire qu'il s'agit d'une conversion de modèle d'un processus d'assemblage : le passage *d'un modèle plutôt analytique*, plus détaillé, à *un modèle synthétique*, plus compact, en conservant la précision du modèle de départ. De cette façon, nous assurons l'exploitation des avantages du premier sur la structure équivalente, mais plus "souple", du deuxième.

Nous avons défini l'objectif de notre recherche :

> Obtenir les graphes de précédence à partir d'un ensemble de gammes d'assemblage.

Le graphe de précédence est **le graphe de la relation de précédence**, qui est à son tour une relation d'ordre partiel définie sur l'ensemble des tâches d'assemblage. En utilisant la classification et la décomposition, nous avons montré que notre problème est ramené à la détermination des graphes de précédence linéaires qui correspondent à *un ensemble de gammes d'assemblage linéaires contenant la même tâche de chargement*.

Les contributions originales de ce chapitre concernent la résolution du problème susmentionné.

Pour formuler mathématiquement ce problème, nous avons défini l'**équivalence** d'un graphe de précédence à un ensemble de gammes. Nous avons formulé et démontré **une condition nécessaire et suffisante** pour que cette équivalence soit remplie :

> Un ensemble de gammes d'assemblage est représenté de façon biunivoque par un graphe de précédence si et seulement si il possède la propriété Π.

Cette propriété reflète une spécificité de la relation de précédence, caractérisée par l'existence des *gammes d'assemblage symétriques* et, ensuite, par **une forme unitaire d'écriture des gammes**, mise en évidence à l'aide de la nouvelle notion de "**relation d'indifférence**" – définie par complémentarité avec la relation de précédence – et du graphe de cette relation – **le graphe d'indifférence**.

Lorsqu'un ensemble de gammes ne remplit pas la propriété Π, il n'est pas représenté par un seul graphe, mais par *un ensemble équivalent de graphes de précédence*.

L'exploitation des résultats obtenus a eu comme objectif la conception d'*algorithmes* qui résolvent le problème de l'obtention de **l'ensemble minimal** de graphes de précédence **équivalent** à un ensemble de gammes. Afin d'arriver à ce but, les algorithmes ont été fondés sur l'idée de trouver *les sous-ensembles maximaux qui ont la propriété* Π.

IV METHODES DE CONCEPTION DES SYSTEMES D'ASSEMBLAGE – AFFECTATION DES TACHES ET EQUILIBRAGE

IV.1 Introduction. Généralités

IV.1.1 Classes de problèmes à résoudre

Dans [GHOS 89] on nous suggère que la conception – ou la structuration – des systèmes d'assemblage est pratiquement réduite au problème de l'équilibrage des lignes, mais qu'elle reste néanmoins difficile à traiter dans une démarche constructive de synthèse. Nous allons essayer d'abord d'encadrer la problématique de la conception des systèmes d'assemblage au sein de la problématique générale de l'assemblage, en soulignant que *la conception* est essentiellement *une démarche de synthèse*. La présentation qui suit est pour l'instant non formelle et intuitive.

Nous pouvons regarder la problématique concernée par l'assemblage comme étant formée d'un seul grand problème, dont le but général est la conception des systèmes d'assemblage, soumise ou non aux critères d'optimum. Dans les grandes lignes, ce grand problème est lui-même un problème de synthèse. Il comporte une décomposition en sous-problèmes concernant les étapes successives de cette synthèse, en s'appuyant également sur des démarches d'analyse. Rappelons, par exemple, **la modélisation** des produits à assembler et la modélisation des processus d'assemblage, qui font l'objet d'un traitement détaillé dans les chapitres antérieurs. Le but de la modélisation est l'obtention d'un modèle ; de ce point de vue, nous pouvons dire que la modélisation a également un caractère synthétique. Mais l'étude des aspects temporels des systèmes d'assemblage est entièrement analytique.

La démarche de **conception** effective se construit autour de l**'affectation des tâches** aux postes de travail. Ce qui représente en principal une action de **découpage en postes de travail**, réalisée sur un modèle du processus d'assemblage – le graphe d'assemblage ou bien le graphe de précédence. L**'équilibrage** est une **affectation optimale**, réalisée *seulement* sur le graphe de précédence. L'équilibrage en temps réel s'appelle équilibrage dynamique.

Une fois que le système d'assemblage est conçu, le problème de conception pourrait être considéré comme résolu. Mais un peu d'attention nous montre qu'il reste encore le problème du comportement ultérieur du système, qui fait l'objet du **pilotage** des systèmes

d'assemblage. Par pilotage nous imposons quelques performances au comportement d'un système avec une configuration déjà connue. Il est logique que ces spécifications soient prises en compte dès la phase de conception, si possible. Sinon, il s'agit de résoudre le problème de la **restructuration** (reconfiguration) des systèmes d'assemblage, qui implique la **réaffectation des tâches** et éventuellement un nouveau ordonnancement des produits. Il en résulte que la conception doit se faire d'une manière qui assure la plus facile restructuration. À ce sujet, nous parlons de la *réactivité* des systèmes d'assemblage, aptitude à la reconfiguration lors de pannes ou de variations de charge.

L'encadrement des problèmes de la conception des systèmes d'assemblage dans la problématique générale de l'assemblage est illustré au tableau ci-dessous.

PROBLÈME GÉNÉRAL	SOUS-PROBLÈME	Brève description	Analyse	Synthèse
Modélisation	Modélisation des produits à assembler	- Élaboration des opérations et des gammes d'assemblage à partir des prototypes des produits	•	•
	Modélisation des processus d'assemblage	- Obtention des arbres/graphes d'assemblage ou des graphes de précédence	•	•
	Analyse des aspects temporels d'un poste de travail	- Calcul du temps de cycle	•	
	Analyse des aspects temporels d'un système d'assemblage	- le régime périodique stabilisé du système vu comme système à événements discrets dans les dioïdes ($\mathbf{R}\cup\{-\infty\}$, max, +) ou ($\mathbf{R}\cup\{-\infty\}$, min, max)	•	
Conception	Découpage en postes de travail	- affectation des tâches aux postes de travail		•
	Équilibrage des lignes d'assemblage	- affectation *optimale* des tâches aux postes de travail		•
	Restructuration (reconfiguration) des systèmes d'assemblage	- réaffectation des tâches en cas d'une défaillance - nouveau ordonnancement des produits		•
Pilotage	Pilotage réactif	- pilotage d'un système conçu en utilisant son potentiel de réactivité		•
	Équilibrage *dynamique* des lignes d'assemblage	- réaffectation optimale des taches *en temps réel* à l'aide d'algorithmes performants		•

Fig. IV-1. La conception des systèmes d'assemblage dans le cadre de la problématique de l'assemblage

La difficulté du problème de conception réside d'une part dans le fait que certaines contraintes, imposées au début, ne peuvent être vérifiées que dans les dernières étapes de la conception. Par exemple, le temps de cycle ne peut être calculé que lorsque la structuration du système a été établie et que les ressources ont été affectées [MÎNZ 95]. D'autre part, dans une démarche de conception dans l'esprit de la méthode L.A.B., nous devons prendre en compte toutes les variantes possibles de structure et d'affectation de ressources. Ce qui conduit à l'explosion combinatoire du nombre des solutions possibles.

IV.1.2 Affectation et réaffectation des tâches dans un système d'assemblage

La résolution du problème de l'affectation des tâches aux postes de travail repose sur l'obtention d'un *découpage en postes*. Le découpage est une partition sur l'ensemble de tâches, dont chaque sous-ensemble a la *structure d'un poste de travail* au sens de la définition opératoire trouvée dans [MÎNZ 95]. Cette définition établit qu'un poste de travail est déterminé par l'ensemble de tâches qu'il exécute et complété par la spécification des équipements désignés pour accomplir ces tâches et par leurs séquences opératoires. La structure de poste de travail est définie par rapport au graphe de précédence.

La formulation du problème de découpage, telle que nous allons la présenter dans le deuxième paragraphe de ce chapitre, émerge de la formulation du problème d'équilibrage des lignes, en relaxant certaines contraintes et en supprimant tout critère d'optimalité. Cette observation exprime en d'autres mots que l'équilibrage est en fait une affectation optimale.

L'étude et la résolution du problème de la réaffectation s'imposent lorsque nous admettons l'occurrence potentielle des *défaillances* dans un système d'assemblage existant. Ces défaillances imposent qu'une autre configuration du système soit trouvée. Il s'agit en principe *d'affecter d'une autre façon les tâches* aux postes. S'il existe une telle nouvelle configuration, le système va fonctionner *en mode dégradé*, de manière sous-optimale, puisque le temps de cycle deviendra forcément plus grand.

Donc, nous constatons que le problème de la réaffectation est une reprise du problème de découpage au niveau du système modifié, qui est pourtant affecté d'une combinatoire plus réduite. Dans [MÎNZ 95] on décrit une approche de la réaffectation des tâches par rapport aux défaillances des opérateurs dans un système d'assemblage.

Le problème de la réaffectation des tâches est le point de départ pour les problèmes impliqués par l'équilibrage et le pilotage des systèmes d'assemblage.

IV.1.3 L'équilibrage comme problème d'affectation optimale

L'optimisation du fonctionnement des lignes d'assemblage est un problème important de la production de masse, dont la complexité a justifié les efforts de recherche de méthodes de résolution de plus en plus efficaces. La littérature a consacré le nom de "équilibrage" pour désigner les approches d'optimisation des lignes. Le terme anglais est "Assembly Line Balancing", d'où provient l'acronyme ALB. Le premier qui a exprimé mathématiquement les exigences de ce problème est Salveson, en 1955. Sa démarche a été continuée en termes stochastiques par Moodie et Young en 1965. Ultérieurement, les méthodes heuristiques et la programmation dynamique ont été constamment défiées par "la course aux dimensions", c'est-à-dire l'accroissement exponentiel du nombre des solutions possibles.

Une ligne d'assemblage "équilibrée" est une ligne dont le découpage en postes remplit un certain critère d'optimum. En ce qui concerne la définition des critères d'optimum pour le problème d'équilibrage, l'évolution chronologique des approches montre un consensus des auteurs : les aspects temporels de l'activité d'assemblage sont les plus importants. Tous les chercheurs considèrent en tant qu'objectif principal *la minimisation du temps de cycle*, mais il existe plusieurs méthodes pour calculer ce temps. À cet objectif s'ajoute habituellement l'obtention *des chargements presque égaux pour tous les postes de travail* de la ligne, puisque les affectations possibles des tâches ont des degrés différents d'équilibrage. Nous disons qu'une ligne qui remplit les deux conditions s'appelle "optimalement équilibrée".

Parfois, un mode équivalent pour caractériser un système parfaitement équilibré est la distribution aussi uniforme que possible du temps d'inactivité aux postes de travail, afin que *le délai global* de la ligne d'assemblage *soit minimisé*. Une autre formulation peut être donnée en tenant compte de *la probabilité d'arrêt du système*, qui doit aussi être *minimisée*. Ces deux

dernières approches sont des reprises des formulations données par Moodie et Young en 1965, respectivement par Reeve en 1971. Une étude comparative entre les deux approches est rapportée dans [SURE 96]. La définition formelle du problème d'équilibrage, donnée dans le deuxième paragraphe de ce chapitre, utilise comme critère d'optimum *la minimisation du temps total d'inactivité* ("total idle time" en anglais) des postes de travail.

IV.1.4 Le pilotage des systèmes d'assemblage

Le pilotage des systèmes d'assemblage est un problème lié à la restructuration, lorsqu'il s'applique aux systèmes soumis aux pannes et dysfonctionnements des ressources, c'est à dire aux *perturbations internes*. Afin de résoudre ce problème, il faut développer une *démarche de conception systémique* : il s'agit de concevoir un nouveau système – *le système de pilotage*. Selon sa description générale, trouvée dans la thèse [YIN 95], un système de pilotage est un système de commande hiérarchisée qui poursuit la réalisation d'un objectif de production pendant une période relativement courte, par l'intermédiaire de la commande et du contrôle de la partie opératoire du système de production.

Physiquement, le système de pilotage se trouve dans la proximité du système piloté. Ce dernier doit donc avoir une bonne *flexibilité* et un grand degré de *réactivité*. Un sens élargi de cette dernière notion est l'adaptation temporelle aux changements de l'environnement, c'est à dire aux *perturbations externes*. Plus précisément, la réactivité d'un système désigne la capacité d'autocontrôle et de réaction en temps réel aux événements perturbateurs par décisions qui tendent à sauvegarder le contexte et la sécurité du système d'exécution et à continuer l'activité sans affecter sa qualité [YIN 95].

Dans [MÎNZ 95] on trouve la relation de calcul pour *le potentiel de réactivité* d'un découpage en postes d'un système d'assemblage. Cette relation peut être vue comme une particularisation restreinte, mais exacte, de la notion de "réactivité" au cas des systèmes d'assemblage pilotés. Le potentiel de réactivité en cas de panne peut constituer un critère d'évaluation des découpages issus d'un algorithme d'affectation des tâches. Il s'agit de juger les solutions potentielles selon un critère d'exploitation du système d'assemblage – en particulier, cette exploitation peut envisager le pilotage du système.

Dans [YIN 95] on trouve une approche hiérarchisée de pilotage réactif des systèmes d'assemblage en anneau, principalement orientée vers la reconception de l'ordonnancement à court et à très court terme.

IV.1.5 Objectif

Après avoir passé en revue les aspects concernés par la problématique de la conception des systèmes d'assemblage, nous allons concentrer notre attention sur **le problème d'équilibrage des lignes comme problème de découpage (affectation) optimale des tâches aux postes de travail**. Au cours de cette introduction nous avons montré que la formulation de ce problème utilise le modèle de type graphe de précédence pour le processus d'assemblage. Lors de ce que nous avons présenté dans le troisième chapitre de ce travail, nous connaissons maintenant une méthode systématique d'obtention des graphes de précédence à partir d'un ensemble de gammes d'assemblage.

Comme première étape de la résolution du problème d'équilibrage, nous allons donner les descriptions générales des **algorithmes de découpage en postes de travail**, aussi bien pour le cas monoproduit, que pour le cas multiproduit [MÎNZ 95].

Le problème d'équilibrage comporte également une **formulation systémique**, appropriée à la résolution par programmation dynamique discrète.

En fin de ce chapitre nous allons essayer une synthèse des méthodes de résolution du problème d'équilibrage, dont le point principal est l'utilisation des différentes techniques d'optimisation. L'équilibrage des lignes d'assemblage est essentiellement un problème

d'optimisation combinatoire, naturellement lié à la *recherche heuristique*. Les approches rencontrées dans la littérature regardent aussi bien les méthodes "classiques", que les méthodes plus récentes, comme, par exemple, l'optimisation par *algorithmes génétiques* ou par *recuit simulé*. Le principe des *méthodes stochastiques* d'optimisation sera aussi présenté.

IV.2 Approches du problème d'équilibrage des lignes d'assemblage

IV.2.1 Formulation du problème d'équilibrage des lignes d'assemblage

IV.2.1.1 Le cas déterministe monoproduit

La formulation du problème d'équilibrage des lignes d'assemblage, présentée ensuite, est valable également pour une ligne de fabrication quelconque.

Soit S l'ensemble des tâches d'assemblage exécutées dans un système d'assemblage et soit G=(S, U), $U \subseteq S \times S$, un graphe de précédence associé à S (conformément à la définition III-3). Les éléments suivants décrivent le probléme d'équilibrage des lignes d'assemblage dans ***le cas déterministe monoproduit***.

On donne :
- le graphe de précédence G=(S, U), où $S = \{s_1, s_2, ..., s_n\}$ est l'ensemble des tâches d'assemblage ;
- l'application $t : S \rightarrow \mathbf{R}^+$, qui donne les temps *déterministes* d'exécution des tâches ;
- le temps de cycle T du système d'assemblage, qui satisfait les conditions suivantes : $\{T \geq t(s_i)\}_{i=1,...,n}$

On demande :
- la famille d'ensembles $F^* = \{W_1, W_2, ..., W_m\}$, $W_i \subseteq S$, $1 \leq i \leq m$, telle que :

(IV.1) $$\bigcup_{i=1}^{m} W_i = S \ ;$$

(IV.2) $$W_i \cap W_j = \varnothing, \ 1 \leq i < j \leq m \ ;$$

(IV.3) $$\forall (a,b) \in U, \ a \in W_i, \ b \in W_j \ \Rightarrow \ i \leq j \ ;$$

(IV.4) $$t_{cycle}(W_i) = t(W_i) = \sum_{x \in W_i} t(x) \leq T, \ 1 \leq i \leq m \ ;$$

(IV.5) $$T^-(F^*) = \min_F \{ T^-(F) \mid F \text{ remplit les conditions (IV.1} \div \text{4)} \},$$

où $T^-(F) = m \cdot T - \sum_{i=1}^{n} t(s_i)$ est le temps total qui n'est pas utilisé par la famille F.

Les conditions (IV.1) et (IV.2) montrent que la famille F^* est une *partition* de l'ensemble S, ce qui est équivalent à l'affectation de chaque tâche à un seul poste de travail.

La nécessité de l'accomplissement des relations de précédence est exprimée par la condition (IV.3) : deux tâches qui se précèdent doivent être affectées respectivement à deux postes qui se précèdent sur la ligne d'assemblage, ou bien au même poste.

La condition (IV.4) demande que le temps de cycle de tout poste, défini comme somme des temps des tâches composantes, ne dépasse pas le temps de cycle, T, imposé au système.

Cette définition du temps de cycle d'un poste montre l'existence d'une hypothèse supplémentaire : dans tout poste travaille *un seul opérateur*.

La condition (IV.5) nous permet d'observer que l'équilibrage doit s'obtenir lors de *la minimisation du temps total d'inactivité* de la ligne.

En ce contexte, nous pouvons remarquer que la résolution du problème d'équilibrage implique dans une première étape la résolution du ***problème de découpage en postes***, ou, autrement dit, du problème d'affectation des tâches aux postes de travail. La définition de ce problème contient les mêmes éléments susmentionnés, en relaxant la contrainte de l'existence d'un seul opérateur dans tout poste et en renonçant à la contrainte d'optimalité. Ce qui signifie que :

- les conditions (IV.1) et (IV.2) restent inchangées (voir (IV.6) et (IV.7) ci-dessous) ;
- la condition (IV.5) est complètement éliminée ;
- la condition (IV.4) est modifiée (voir (IV.9) ci dessous).

De plus, au lieu de la condition (IV.3) nous introduisons la condition (IV.8), plus forte, représentant *la contrainte de structure*, dont la formulation impose implicitement l'accomplissement des relations de précédence. La signification exacte de la contrainte structurelle sera donnée dans le sous-paragraphe IV.2.2.

Ainsi, tout en gardant les notations ci-dessus et les données d'entrée, une famille $F = \{W_1, W_2, ..., W_m\}$, $W_i \subseteq S$, $1 \le i \le m$, représente un ***découpage en postes*** si elle remplit les conditions suivantes :

(IV.6) $\bigcup_{i=1}^{m} W_i = S$;

(IV.7) $W_i \cap W_j = \varnothing,\ 1 \le i < j \le m$;

(IV.8) les ensembles W_i, $i=1, ... m$ ont la structure d'un poste de travail par rapport aux graphe G ;

(IV.9) $t_{cycle}(W_i) = t(W_i) \le T,\ 1 \le i \le m$;

IV.2.1.2 Le cas multiproduit

La formulation, et aussi bien la résolution du problème d'équilibrage dans le cas multiproduit, peuvent être comprises par *analogie* avec le cas monoproduit. Les remarques faites auparavant au sujet de la liaison entre les deux problèmes de structuration – l'équilibrage et l'affectation des tâches aux postes de travail (le découpage en postes) – restent valables. Nous allons détailler ce sujet plus loin.

Le problème d'équilibrage ***multiproduit*** est décrit par les éléments suivants ([THOM 70], [MACA 72]).

On donne :

- les graphes de précédence $G_k=(S_k, U_k)$, $k=1,2,...K$, qui modèlent l'assemblage d'un ensemble de K produits, où S_k est l'ensemble des tâches nécessaires pour obtenir le produit k ;
- l'ensemble de n tâches *distinctes* nécessaires pour réaliser les K produits :

(IV.10) $S = \bigcup_{k=1}^{K} S_k = \{s_1, s_2, ...s_n\}$

et l'ensemble :

(IV.11) $U = \bigcup_{k=1}^{K} U_k$;

- l'application $t : S \times \{1,2,...K\} \rightarrow \mathbf{R}^+$, qui donne les temps *déterministes* d'exécution de toute tâche pour tout produit :

$$t(s_i,k) = \begin{cases} t(i,k) \neq 0 - \text{le temps d'exécution de la tâche } i \text{ pour le produit } k \\ 0, \text{ si le produit } k \text{ ne nécessite pas la tâche } i \end{cases}$$

- la période de temps T, qui caractérise la *rythmicité* du système, pendant laquelle nous voulons réaliser N_k exemplaires du produit k, $k=1,2,...K$.

On demande :

- la famille d'ensembles W^* et les nombres entiers $p(i,k,j)$ – chacun d'eux exprimant le nombre d'exemplaires du produit k pour lesquels la tâche i est exécutée dans le poste j – qui doivent remplir les conditions suivantes :

(IV.12) $W^* = \{W_1^*, W_2^*, \ldots, W_m^*\}, \; W_j^* \subset S, \; j = 1,2,...m$;

(IV.13) $S = \bigcup_{j=1}^{m} W_j^*$, où les ensembles W_j^* correspondent aux postes de travail ;

(IV.14)
$$\begin{cases} \forall q \in \{1,2,...m\}, \; \forall y \in W_q^*, \\ \forall x \in \{z \mid \exists k \in \{1,2,...K\}, (z,y) \in U_k\}, \\ \forall r \in \{j \mid x \in W_j^*\}, \; \overline{|r \leq q|} \end{cases}$$

(IV.15) $\sum_{j=1}^{m} p(i,k,j) = N_k$;

(IV.16) $t_{cycle}(W_j^*) = \sum_{x \in W_j} \sum_{k=1}^{K} t(x,k) \cdot p(x,k,j) \leq T, \; j = 1,2,...m$;

(IV.17)
$$\begin{cases} \forall x \in S \text{ avec } x \in W_{j_1}^*, ...W_{j_{u(x)}}^*, \text{ où } u(x) = \text{card}\{j \mid x \in W_j^*\} > 1 \\ j_{h+1} = j_h + 1, \quad h = 1,2,...u(x) - 1 \end{cases} ;$$

(IV.18) $T_N(W^*) = \min_{W} \{T_N(W) \mid W \text{ remplit les conditions (III.12} \div \text{17)}\}$

(T_N est le temps total d'inactivité)

Les conditions (IV.12) et (IV.13) expriment le fait que la famille W^* est une *couverture* (ou recouvrement) de S et *non pas une partition*, comme dans le cas de l'équilibrage monoproduit, ce qui permet la multi-affectation. Nous remarquons également la présence d'une nouvelle donnée d'entrée par rapport au problème d'équilibrage monoproduit : la période de temps T. Néanmoins, cette donnée peut être vue comme *temps de cycle* du système, mais avec une signification plus nuancée. Ainsi, le temps de cycle T désigne la *cadence* du système multiproduit, lorsqu'il répond à une *contrainte de volume de la production* : nombre d'exemplaires produits pour chaque membre de la famille de produits dans une certaine période de temps.

Nous pouvons dire que le modèle élémentaire du processus d'assemblage pour toute la famille de produits est toujours *un graphe de précédence*, noté par G = (S, U), où S et U sont respectivement définis par les relations (IV.10) et (IV.11). Il en résulte que ce graphe s'obtient comme *union* des graphes de précédence de tous les produits.

Une solution du problème d'équilibrage multiproduit contient aussi les nombres entiers *p(i,k,j)*, soumis aux conditions (IV.14÷17). La condition (IV.14) exprime le fait que pour toute tâche *y* d'un poste de travail W_q^*, si *x* est un prédécesseur quelconque de *y* dans un graphe G_k, alors les indices *r* de tous les postes de travail qui contiennent *x* doivent être inférieurs à *q*. Ceci est possible si le graphe G = (*S*, *U*) est *acyclique*.

La condition (IV.15) impose que le nombre total d'exécutions de la tâche *i* pour le produit *k* soit N_k.

Le temps de cycle de chaque poste doit être inférieur au temps de cycle *T* du système multiproduit, ce qui s'exprime par la condition (IV.16). L'expression de calcul de ce temps de cycle émerge, comme dans le cas monoproduit, de l'hypothèse supplémentaire de l'*opérateur unique* qui travaille au sein de chaque poste de travail du système multiproduit. Sous cette hypothèse, le temps de cycle d'un poste est la somme des durées d'exécution de toutes les tâches composantes, pour tous les exemplaires des produits qui nécessitent ces tâches.

La condition (IV.17) nous montre que toute tâche doit être exécutée, pour tous les produits et tous les exemplaires, dans des postes consécutifs.

La condition (IV.18) représente le critère d'optimum. Nous allons voir que toutes les autres conditions, avec changements mineurs, existent également dans la formulation du *problème du découpage* en postes *multiproduit*. C'est ainsi que l'équilibrage est dans ce cas, comme dans le cas monoproduit, un problème d'affectation optimale. Dans les grandes lignes, une solution *W* du problème d'affectation (du découpage) multiproduit remplit les conditions (IV.12÷17). Une telle solution devient solution du problème d'équilibrage – notée *W** – si elle minimise *le temps total qui n'est pas utilisé*, noté par $T_N(W^*)$ et nommé aussi *temps total d'inactivité*.

Une signification plus profonde de ce critère d'optimum peut être saisie en déduisant l'expression de calcul du temps total d'inactivité d'une solution *W*, appartenant à la famille des solutions qui satisfont aux conditions (IV.12÷17).

Notons par $T(W_j)$ *le temps de travail effectif* pour le poste W_j de *W*, *j=1,2,...m*. Il est en fait *le temps de cycle du poste* W_j, qui satisfait (IV.16). Il résulte que le temps d'inactivité du poste W_j est $T\text{-}T(W_j)$. Évidemment, il est vrai que :

$$T_N(W) = \sum_{j=1}^{m}(T - T(W_j)) = m \cdot T - \sum_{j=1}^{m} T(W_j),$$

expression où nous utilisons ensuite (IV.16). Nous obtenons successivement :

$$\begin{aligned} T_N(W) &= m \cdot T - \sum_{k=1}^{K}\left[\sum_{j=1}^{m}\left(\sum_{x \in W_j} t(x,k) \cdot p(x,k,j)\right)\right] \\ &= m \cdot T - \sum_{k=1}^{K}\left[\sum_{j=1}^{m}\left(\sum_{x \in W_j \cap S} t(x,k) \cdot p(x,k,j)\right)\right] \\ &= m \cdot T - \sum_{k=1}^{K}\left[\sum_{x \in S_k} t(x,k) \cdot \left(\sum_{j=1}^{m} p(x,k,j)\right)\right] \end{aligned}$$

En utilisant la condition (IV.15), nous obtenons ensuite :

$$T_N(W) = m \cdot T - \sum_{k=1}^{K}\left(\sum_{x \in S_k} t(x,k) \cdot N_k\right) \qquad \text{(IV.19)}$$

Nous remarquons que l'accomplissement du critère d'optimum (IV.18) par une affectation (un découpage) *W*, qui remplit également la condition (IV.19), conduit à une solution pour laquelle ***le nombre des postes de travail est minimum***, puisque N_k et *t(x,k)* sont

des constantes. On peut prouver que le problème d'équilibrage multiproduit admet au moins une solution [THOM 70].

Comme nous avons déjà observé, en renonçant à la contrainte d'optimalité et à l'hypothèse de l'existence d'un seul opérateur dans chaque poste, nous obtenons l'ensemble des conditions définissant ***le problème de découpage en postes d'un système d'assemblage multiproduit***. Cela signifie que la condition (IV.18) n'existera plus et que la condition (IV.16) sera modifiée en éliminant l'expression de calcul du temps de cycle. Nous obtenons ainsi une plus grande généralité, bien que la contrainte de l'opérateur unique est vérifiée en réalité, dans la plupart des postes de travail. Comme suite, la condition (IV.16) devient :

(IV.20) $t_{cycle}(W_j) \le T,\ j = 1,2,...m$

En gardant les notations faites ci-dessus, les données d'entrée du problème d'équilibrage et les relations vérifiées par elles (IV.10 et IV.11), une solution W du problème de découpage multiproduit est un recouvrement de S qui satisfait aux conditions (IV.12÷15) et (IV.17) – où la notation W^* est remplacée par la notation W – et à la condition (IV.20).

Une discussion s'impose au sujet de l'accomplissement de la condition (IV.14). Les observations que nous allons faire permettent d'établir une *analogie* entre le cas monoproduit et le cas multiproduit du problème de découpage. Nous avons vu que la condition susmentionnée est remplie si le processus d'assemblage de la famille de produits est modélisé par un *graphe de précédence* G = (*S*, *U*) *acyclique*.

Dans [MÎNZ 95] on montre que cela est possible sous l'hypothèse que la famille de produits accepte *au moins une gamme générique*, c'est à dire un arbre d'assemblage générique. Ces notions sont l'expression de la généralisation des notions bien connues, mais définies seulement pour le cas monoproduit : composant, solidarisation et caractère complémentaire. Cette généralisation devient possible dès que nous supposons que la famille est assemblée dans le même système d'assemblage, puisqu'elle devient ainsi *une famille technologique*. Il devient également possible d'établir des *relations de correspondance* entre les groupes fonctionnels de composants, et aussi entre les solidarisations et entre les caractères complémentaires ([DUFR 91]). Nous pouvons dire qu'il s'agit d'*une famille iso-structurelle* de produits (voir également I.2.2.2).

Chaque produit de la famille peut être modélisé par un arbre d'assemblage dérivé de l'arbre d'assemblage générique. Chaque arbre est ensuite transformé en un graphe de précédence (comme nous l'avons montré au cours du paragraphe II.2 du deuxième chapitre). Nous obtenons ainsi l'ensemble de graphes de précédence $\{G_k\}_{k=1,...K}$, dont l'union est le graphe G. Pratiquement, ce graphe est la transformation de l'arbre d'assemblage générique, qui est toujours acyclique.

Nous avons montré que le problème d'équilibrage peut être décomposé en ***deux sous-problèmes*** : le problème du découpage en postes et le problème d'optimisation.

Le travail cité ci-dessus présente des algorithmes de résolution du problème de découpage en postes. En utilisant l'analogie que nous venons de présenter, le traitement du cas multiproduit est réduit au cas monoproduit. Le sous-paragraphe suivant est dédié à la description de ces algorithmes.

IV.2.2 Algorithmes de découpage en postes de travail [MÎNZ 95]

IV.2.2.1 Le cas monoproduit

Le problème de découpage monoproduit, inspiré du problème d'équilibrage monoproduit, utilise comme modèle du processus d'assemblage le *graphe de précédence*.

L'approche proposée dans la thèse [MÎNZ 95] repose sur l'utilisation du *graphe d'assemblage*, qui est un graphe de précédence construit d'une manière systématique, permettant une évaluation plus correcte des temps d'exécution des tâches.

Nous avons montré dans le chapitre antérieur que le graphe de précédence peut être construit d'une manière systématique à partir d'un ensemble de gammes d'assemblage.

Le point de départ de l'algorithme de découpage est la génération des ***ensembles de tâches qui ont la structure de poste de travail***. Cette procédure sera détaillée plus loin. Elle repose sur le fait qu'un *poste de travail* se définit *par rapport au graphe de précédence*.

L'appellation de **poste candidat** sera utilisée pour un ensemble de tâches qui satisfait à toutes les contraintes imposées par la définition d'un poste de travail et qui a été vérifié jusqu'à l'étape courante de la conception.

Nous remarquons facilement que le découpage en postes peut être trouvé *d'une manière récursive* : nous choisissons un ensemble de tâches, qui correspondent à un poste de travail et ensuite nous trouvons un découpage pour le reste des tâches. Mais cette stratégie est inefficace, parce que la nature du problème engendre l'explosion combinatoire des solutions potentielles. Ce fait conduit à la nécessité d'utiliser une autre stratégie, plus efficace, qui comporte ***deux étapes*** :

- générer *tous les postes de travail candidats* qui remplissent les contraintes imposées par le problème de découpage en postes ;
- trouver *toutes les partitions du graphe de précédence* utilisant les postes générés dans l'étape antérieure.

La deuxième étape utilise toujours une procédure récursive, mais, cette fois-ci, les postes sont choisis dans un ensemble plus réduit.

L'efficacité de cette stratégie peut être encore augmentée en utilisant les contraintes au fur et à mesure du développement des postes candidats. La représentation interne d'un poste candidat s'enrichit au cours de son traitement, comme suite à l'évolution de sa description : d'un ensemble de tâches vers l'ensemble d'éléments figurant dans la définition d'un poste de travail.

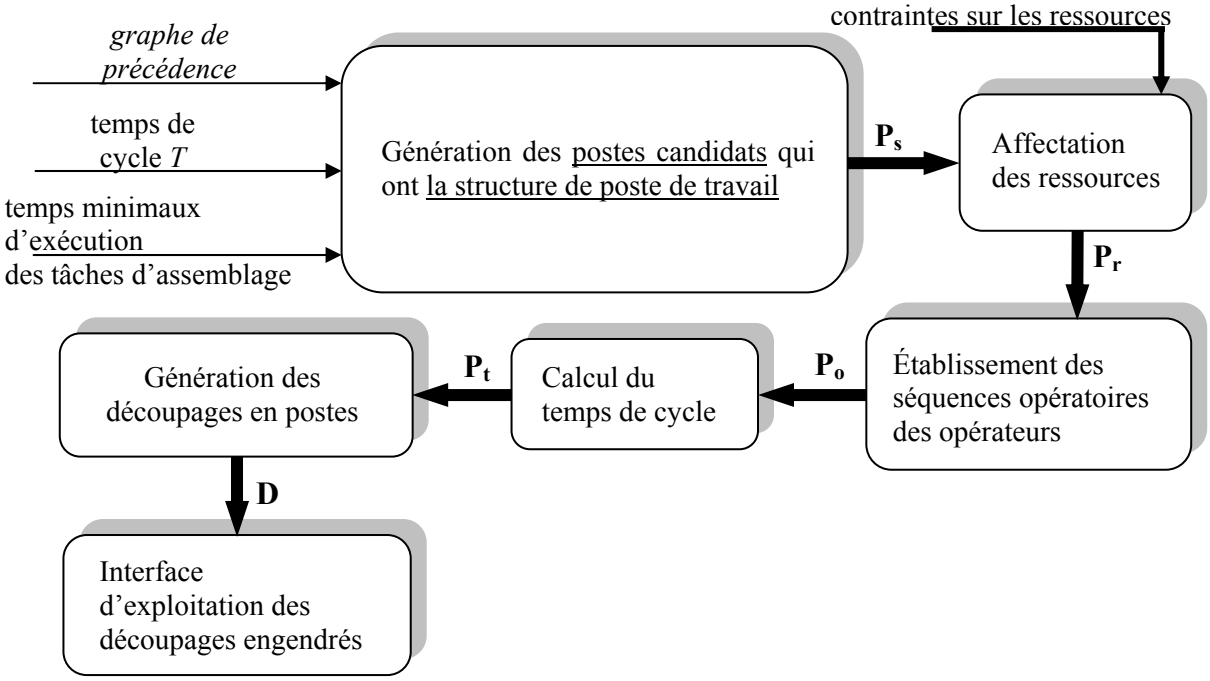


Fig. IV-2. La structure de l'algorithme de découpage en postes

À la figure IV-2 nous présentons la structure de l'algorithme de découpage en postes. Les quatre premières étapes établissent les postes candidats, en utilisant les informations fournies par le concepteur. Les ensembles de postes candidats engendrés à la fin de chaque étape de traitement sont notés respectivement par $\mathbf{P_s}$, $\mathbf{P_r}$, $\mathbf{P_o}$, $\mathbf{P_t}$. À partir de ces postes candidats, l'étape suivante génère l'ensemble des découpages possibles, noté par **D**. La dernière étape est une interface interactive d'exploitation des découpages engendrés, qui répond aux requêtes du concepteur concernant les affectations de ressources, séquences opératoires, calcul du temps de cycle, etc., afin de choisir "le meilleur" découpage.

Chaque passage d'une étape de l'algorithme à la suivante implique un traitement des *données supplémentaires*.

L'ensemble $\mathbf{P_s}$ issu de la première étape de l'algorithme contient les postes candidats qui remplissent *la contrainte de structure*. Dans cette étape, **un poste candidat** est représenté par **la liste des tâches composantes**.

Dans la deuxième étape, l'algorithme essaye d'affecter un opérateur et un outil à chaque tâche d'un poste candidat. Les nouvelles données traitées sont *les ressources* qui peuvent être affectées. Ainsi la représentation d'un poste candidat s'enrichit de sorte qu'il est décrit par **la liste de ses tâches et des ressources utilisées**. Le cardinal de l'ensemble des postes candidats est soumis à deux tendances opposées : d'une part, certains postes candidats de $\mathbf{P_s}$ seront éliminés – parce qu'il n'existe aucune affectation satisfaisant les contraintes imposées – et d'autre part, certains postes permettent plusieurs affectations. Par suite, dans l'ensemble $\mathbf{P_r}$ issu à la fin de cette étape il y aura des postes qui contiennent les mêmes tâches d'assemblage, mais qui diffèrent du point de vue de la conception, parce qu'ils sont caractérisés par des listes différentes de ressources.

L'établissement des séquences opératoires pour chaque opérateur d'un poste candidat prépare le calcul du temps de cycle du poste. Par rapport à l'étape antérieure, chaque poste candidat de $\mathbf{P_o}$ est caractérisé par une donnée supplémentaire : **à chaque porteur** nous associons **l'ordonnancement des tâches** qu'il accomplit au sein du poste.

La dernière vérification est accomplie par **le calcul du temps de cycle** des postes candidats de $\mathbf{P_o}$. Nous éliminons les postes dont les temps de cycle sont supérieurs au temps de cycle T imposé au système d'assemblage. C'est la seule étape où la diminution du nombre des postes candidats est garantie.

Enfin, nous engendrons l'ensemble **D** de tous les découpages possibles du système d'assemblage en utilisant les postes candidats de $\mathbf{P_t}$.

Maintenant, nous allons nous occuper de ***la contrainte de structure***. Elle intervient dans la première étape de l'algorithme de découpage, où on génère ***l'ensemble des tâches qui a la structure de poste de travail***. L'accomplissement de cette contrainte repose sur les propriétés structurelles du graphe de précédence.

Du point de vue opératoire, un poste de travail est défini par l'ensemble des tâches exécutées pour un seul exemplaire de produit.

> Le graphe de précédence est l'outil le plus général pour décrire les relations de précédence entre les tâches d'assemblage.

Soit S l'ensemble des tâches d'assemblage exécutées dans un système d'assemblage et soit $G=(S, U)$, $U \subseteq S \times S$, un graphe de précédence associé à S (selon la définition III-3).

> Un poste de travail est un ensemble de tâches qui respectent les contraintes de précédence imposées par G.

Les autres contraintes – d'affectation d'équipements, de productivité, coût, etc. – seront éludées pour l'instant.

Notations :

$\mathrm{P}(S)$ – l'ensemble de sous-ensembles de S

$s(x)$ – les successeurs *directs* de $x \in S$

$p(x)$ – les prédécesseurs *directs et indirects* de $x \in S$

$p(X) = \bigcup_{x \in X} p(x)$ - les prédécesseurs directs et indirects des nœuds d'*un ensemble* $X \subset S$

Définition IV-1 : ***poste de travail*** [MÎNZ 95]

> Un ensemble $W \in \mathrm{P}(S)$ a la structure d'un poste de travail par rapport au graphe de précédence G si tout nœud de tout chemin de G, qui joint deux nœuds de W, appartient également à W.

La figure IV-3 suggère les différences entre les ensembles qui ont la structure d'un poste de travail et ceux qui ne l'ont pas.

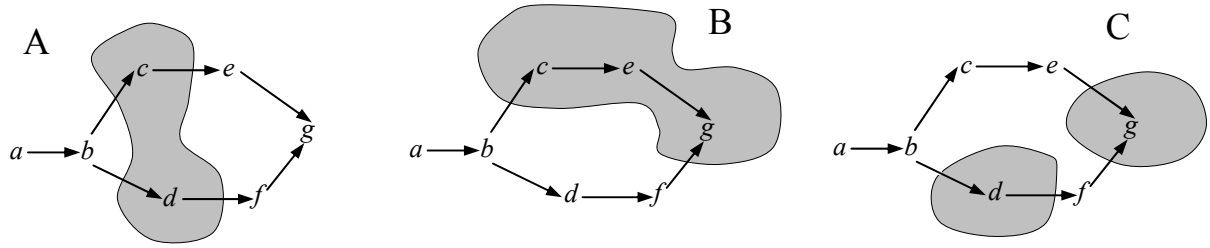


Fig. IV-3. Exemples d'ensemble de tâches qui ont la structure de poste de travail (A, B) et qui ne l'ont pas (C)

Définition IV-2 : ***ensemble admissible*** [MÎNZ 95]

> Un ensemble $X \in \mathrm{P}(S)$ est admissible par rapport au graphe G = (S, U) si $p(X) \subset X$.

Par exemple, dans la figure IV-3, l'ensemble $\{a,b,c,e\}$ est admissible, tandis que l'ensemble $\{c,e,g\}$ ne l'est pas. L'ensemble vide est considéré admissible.

Dans [MÎNZ 95] on trouve la formulation et la démonstration d'*une condition nécessaire et suffisante* pour qu'un ensemble de nœuds du graphe de précédence représente un poste de travail au sens de la définition IV-1.

Proposition P-IV.1 :

> Un ensemble $W \in \mathrm{P}(S)$ a la structure d'un poste de travail par rapport au graphe de précédence G si et seulement si il existe les ensembles E_1, $E_2 \in \mathrm{P}(S)$ *admissibles* par rapport au graphe G, tels que $W = E_1 - E_2$ et $E_2 \subset E_1$.

Remarque :

La conclusion de cette proposition est incluse dans celle d'un théorème connu sur l'équilibrage des lignes, prouvé dans [GUTJ 64]. Les deux résultats sont différents au niveau de leurs hypothèses. La proposition ci-dessus a comme hypothèse la propriété d'un ensemble de tâches d'avoir *la structure de poste de travail* – nous pouvons dire qu'il s'agit d'une propriété "locale" – tandis que l'hypothèse du théorème mentionné est une propriété *globale* d'un ensemble de tâches, de former avec d'autres ensembles une partition de l'ensemble S, qui, de plus, optimise un critère.

Une autre interprétation de cette proposition peut être donnée à l'aide de la notion d'*état du produit intermédiaire*.

Définition IV-3 : ***état du produit intermédiaire*** [BOUJ 84]

> Un état du produit intermédiaire est l'ensemble des caractères fonctionnels du produit déjà établis à un certain instant.

Un *état du produit intermédiaire* correspond à un ensemble de tâches d'assemblage, noté E ($E \subset S$), complètement exécutées. Potentiellement, cet état s'obtient en arrêtant l'exécution des tâches d'assemblage à un certain moment et en marquant les tâches complètement exécutées. Il en résulte aisément que l'ensemble E est *admissible* par rapport au graphe de précédence G.

Par conséquent, la proposition P-IV.1 acquiert une nouvelle signification : *l'ensemble de tâches affectées à un poste de travail doit être la différence entre deux ensembles de tâches, correspondant à deux états du produit intermédiaire*. Ce qui montre l'équivalence entre un poste de travail candidat et un poste de travail déterminé par ALB.

Nous allons montrer ensuite un autre résultat, qui fournit ***une condition suffisante***, mais qui n'est pas nécessaire, pour qu'un ensemble de tâches ait la structure d'un poste de travail. Notre résultat utilise une particularité structurelle du graphe de précédence, mise en évidence sous une forme équivalente au niveau des ***composantes connexes du graphe d'indifférence*** (voir les définitions III-2 et III-8).

Proposition P-IV.2 :

> Soit G_I le graphe d'indifférence associé à un graphe de précédence G. Les composantes connexes de G_I ont la structure d'un poste de travail (par rapport au graphe G).

Démonstration :

Soit C une composante connexe de G_I. Compte tenu de la définition d'un poste de travail, nous devons montrer que tout nœud de tout chemin de G, joignant deux nœuds de C, appartient également à C.

Soit $x, y \in C$. Il peut exister deux situations :

- soit x et y sont *indifférents* et, donc, il n'existe aucun chemin dans G entre eux ;
- soit x et y se trouvent en relation de précédence.

Le premier cas n'est pas intéressant pour notre but. Dans le deuxième cas, les deux nœuds sont liés par au moins un chemin de G. Nous pouvons éliminer aussi la situation où l'un des deux nœuds est le successeur direct de l'autre.

Supposons, par exemple, que $x<y$. Supposons également qu'il existe au moins un nœud "intermédiaire", z, sur un chemin fixé de x à y dans le graphe G. Il en résulte que $x<z<y$. Montrons que $z \in C$.

Nous raisonnons par l'absurde. Supposons que $z \notin C$. Il en résulte que z appartient à une autre composante connexe de G_I : $z \in C'$. Nous utilisons **le lemme L-III.7**, qui dit que *les composantes connexes sont totalement ordonnées* (du point de vue de la relation de précédence entre sous-ensembles de nœuds – voir définition III-9) : soit $C \lhd C'$, soit $C' \lhd C$.

- Si $C \lhd C'$, il s'ensuit que $\forall a \in C, \forall b \in C' : a < b$. Donc, $y<z$, ce qui est une **contradiction**.
- Si $C' \lhd C$, il s'ensuit que $\forall a \in C, \forall b \in C' : b < a$. Donc, $z<x$, ce qui est une **contradiction**.

La supposition faite est fausse, lorsqu'elle nous a conduit à une contradiction. La proposition est montrée.

Malheureusement, la réciproque de la proposition P-IV.2 n'est pas valable : un ensemble de tâches, ayant la structure d'un poste de travail par rapport à un graphe de précédence donné, n'est pas forcément une composante connexe du graphe d'indifférence correspondant. Par exemple, dans la figure IV-3, les ensembles hachurés des cas A et B ont la structure d'un poste de travail, sans être des composantes connexes du graphe d'indifférence.

Pourtant, le dernier résultat peut être utile à la génération ***seulement*** des postes de travail qui correspondent aux composantes connexes du graphe d'indifférence. Pratiquement, cette génération serait équivalente au problème *polynomial* de *trouver les composantes connexes d'un graphe non orienté*. Mais il n'existe aucune raison technologique qui justifie cette manière de réduction implicite du nombre des solutions potentielles.

Dans [MÎNZ 95] la contrainte de structure est traitée par rapport à un graphe d'assemblage. En ce contexte, on a prouvé que le problème de trouver *toutes* les partitions en sous-ensembles qui aient la structure d'un poste de travail a une *complexité polynomiale* si on suppose *une limitation* – réaliste, d'ailleurs – *du nombre de sous-assemblages*. Cette hypothèse se reflète dans le nombre fini des branches du graphe d'assemblage (voir le travail cité).

IV.2.2.2 Le cas multiproduit

Le problème de découpage multiproduit est résolu ***par réduction au cas monoproduit***, en utilisant une technique d'agrégation qui peut être déclinée en deux variantes.

Nous gardons les notations faites au sujet de la formulation du problème de découpage multiproduit (voir IV.2.1.2).

La première variante impose que toutes les N_k exécutions d'une tâche x pour un produit k soient réalisées dans le même poste de travail. Ce cas s'appelle **l'agrégation partielle des durées**. Nous allons introduire la notion de "*tâche agrégée*" et définir les temps d'exécution agrégés. La relation de précédence entre les tâches d'assemblage sera extrapolée aux tâches agrégées. Le problème de découpage multiproduit sera ainsi réduit à un problème de découpage monoproduit sur le graphe G' de la relation de précédence extrapolée.

Dans la deuxième variante, nous imposons que chaque tâche soit complètement exécutée dans un seul poste de travail, pour tous les produits et pour tous les exemplaires de chaque produit. Ce cas s'appelle **l'agrégation totale des durées**, où les temps d'exécution agrégés sont calculés d'une autre manière que dans la première variante. Le problème de découpage multiproduit sera réduit à un problème de découpage monoproduit sur le graphe de précédence G de toute la famille de produits, qui est *l'union* des graphes de précédence de tous les produits (voir les relations IV.10 et IV.11 et l'exemple IV-1 ci-après).

Dans les deux cas, la donnée du temps de cycle imposé, T, du problème original (multiproduit) se conserve au niveau du problème équivalent (monoproduit).

Dans le tableau de la figure IV-4 nous présentons de façon comparative les éléments des deux variantes.

Pour résoudre le problème de découpage monoproduit, nous pouvons appliquer les algorithmes connus. Comme nous l'avons mentionné auparavant, nous pouvons prouver qu'*une condition suffisante* pour que le problème de découpage multiproduit ait des solutions est que le graphe de précédence G soit *acyclique*.

Indépendamment de la variante choisie, une fois que nous obtenons une solution du problème monoproduit, nous construisons la solution du problème original de découpage multiproduit. Nous présentons les détails d'une telle procédure sur un exemple traité aussi bien par la première, que par la deuxième des variantes.

	Agrégation **partielle** des durées (variante 1)	Agrégation **totale** des durées (variante 2)
Hypothèse de base	- toutes les exécutions de chaque tâche pour un certain produit se réalisent dans le même poste de travail : $\forall k \in \{1,2,...K\}, \forall x \in S_k$, $\exists j_0 \in \{1,2,...m\}$ tel que : $x \in W_{j_0}$ et $p(x,k,j_0) = N_k$	- chaque tâche est complètement exécutée dans un seul poste de travail, pour tous les produits et pour tous les exemplaires de chaque produit : $\forall x \in S, \exists j_0 \in \{1,2,...m\}$ tel que : $x \in W_{j_0}$ et $\sum_{k=1}^{K} p(x,k,j_0) = \sum_{h \in \{j \mid x \in S_j\}} N_h$
Condition nécessaire pour l'existence des solutions	(IV.21) $T \geq \max_{k=1,K} \left\{ \max_{x \in S_k} (N_k \cdot t(x,k)) \right\}$	(IV.22) $T \geq \max_{x \in S} \left\{ \sum_{k=1}^{K} (N_k \cdot t(x,k)) \right\}$
Ensemble des *tâches agrégées*	(IV.23) $S' \stackrel{\Delta}{=} \{ s(x,k) \equiv s_{xk} \mid x \in S_k, k = \overline{1,K} \}$	-
Temps d'exécution *agrégés*	(IV.24) $\forall s_{xk} \in S' : t'(x,k) \stackrel{\Delta}{=} t(x,k) \cdot N_k$	(IV.25) $\forall x \in S : t''(x) \stackrel{\Delta}{=} \sum_{k=1}^{K} N_k \cdot t(x,k)$
Modèle du processus d'assemblage : *graphe de précédence*	- l'ensemble des prédécesseurs directs d'un nœud x dans le graphe G_k : $pr(x,k) \stackrel{\Delta}{=} \{ z \mid x,z \in S_k, (z,x) \in U_k \}$ - l'ensemble *total* des prédécesseurs directs d'une tâche x : $P(x) = \bigcup_{k=1}^{K} pr(x,k)$ - l'ensemble des arcs du nouveau graphe de précédence (sur l'ensemble des tâches agrégées) : (IV.26) $U' \stackrel{\Delta}{=} \bigcup_{x \in S} \bigcup_{y \in P(x)} \left\{ \begin{array}{l} (s(y,h),s(x,k)) \mid \\ k,h \in \overline{1,K}, \\ x \in S_k, y \in S_h \end{array} \right\}$ le *nouveau* graphe de précédence G'=(S',U')	G=(S,U) - *l'union* des graphes de précédence de tous les produits $\{G_k=(S_k,U_k)\}_{k=1,2,...K}$
Réduction au *problème de découpage monoproduit* – éléments de définition	- le nouveau graphe de précédence G'=(S', U') - les temps agrégés $\{t'(x,k)\}_{x \in S, k=1,...K}$ - le temps de cycle imposé T	- le graphe de précédence G=(S, U) - les temps agrégés $\{t''(x)\}_{x \in S}$ - le temps de cycle imposé T

Fig. IV-4. Réduction du problème de découpage multiproduit au cas monoproduit – comparaison des deux variantes

Exemple IV-1 :

Nous voulons résoudre le problème de découpage multiproduit pour une famille de trois produits dont les graphes de précédence se trouvent dans la figure IV-5. Le tableau à la figure IV-6 contient les temps d'exécution des tâches et le nombre d'exemplaires pour chaque type de produit (N_k, $k=1,2,3$ – la structure de fabrication).

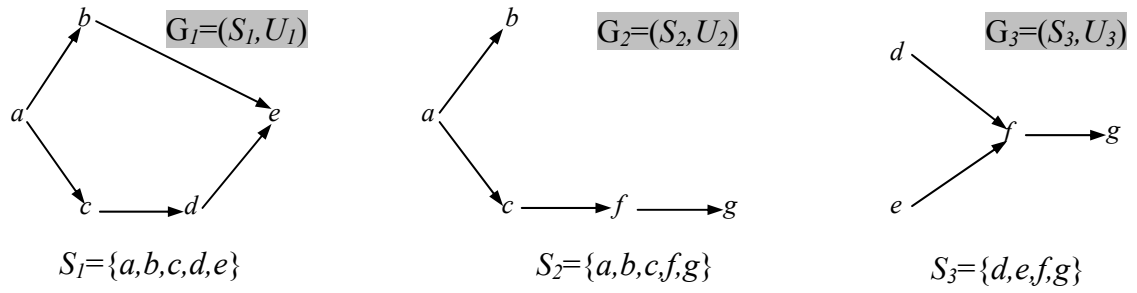


Fig. IV-5. Les graphes de précédence pour une famille de trois produits ($K=3$)

		Durées des tâches						
Produit (k)	N_k	a	b	c	d	e	f	g
1	*5*	2	1	0.2	1	3	0	0
2	*8*	1	2	0.5	0	0	1	0.5
3	*4*	0	0	0	1	4	1.5	1

Fig. IV-6. Les durées des tâches pour un exemplaire de chaque type de produit et la structure de fabrication

Variante 1 *: agrégation partielle des durées*

Supposons que dans ce cas le temps de cycle imposé est $T = 28$ unités de temps.

L'ensemble des tâches d'assemblage est $S = S_1 \cup S_2 \cup S_3 = \{a,b,c,d,e,f,g\}$.

Ensuite, nous construisons l'ensemble des tâches agrégées, conformément à la relation (IV.23) :

$$S' = \{s_{a1}, s_{a2}, s_{b1}, s_{b2}, s_{c1}, s_{c2}, s_{d1}, s_{d3}, s_{e1}, s_{e3}, s_{f2}, s_{f3}, s_{g2}, s_{g3}\}$$

Les temps agrégés $\{t'(x,k)\}_{x\in S,\ k=1,2,3}$, calculés selon la relation (IV.24), sont donnés ci-dessous :

	Tâches agrégées						
Produit (k)	a	b	c	d	e	f	g
1	10	5	1	5	15		
2	8	16	4			8	4
3				4	16	6	4

Fig. IV-7. Le tableau des temps agrégés pour la première variante

Afin de construire le *nouveau* graphe de précédence, G'=(S',U'), qui sert à la formulation du problème de découpage *monoproduit*, équivalent au problème original *multiproduit*, nous devons déduire l'ensemble de ses arcs, U'. Nous construisons les ensembles de tous les prédécesseurs directs de chaque tâche de S, $\{P(x)\}_{x\in S}$:

$P(a) = \varnothing$, $P(b) = \{a\}$, $P(c) = \{a\}$,

$P(d) = \{c\}$, $P(e) = \{b,d\}$, $P(f) = \{c,d,e\}$,

$P(g) = \{f\}$.

Par conséquent, en utilisant la relation (IV.26), il en résulte l'ensemble des arcs du graphe G' de la relation de précédence sur l'ensemble des tâches agrégées. Lorsque les précédences redondantes sont éliminées, le graphe G' se présente comme dans la figure IV-8.

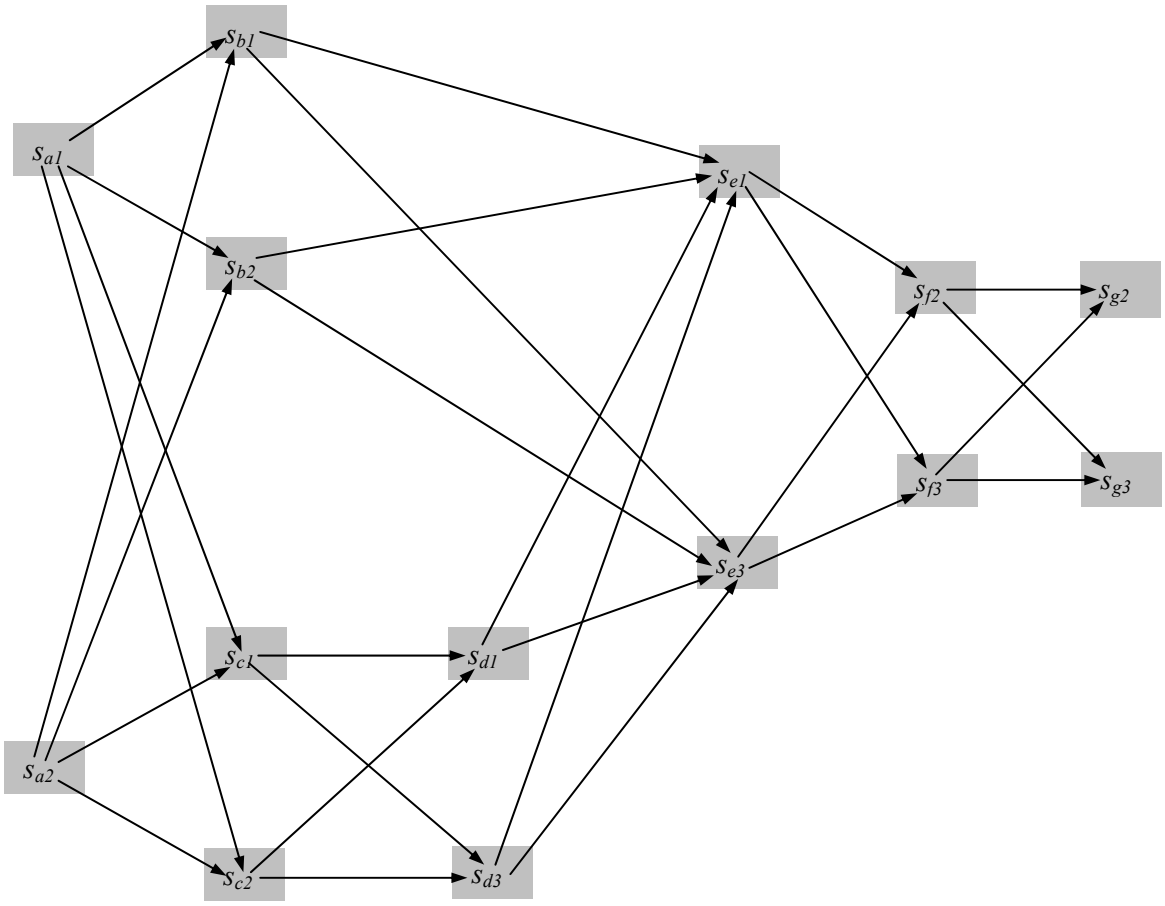


Fig. IV-8. Le graphe de précédence G'=(*S'*,*U'*) de la famille de produits pour la première variante

Une solution possible de ce problème de découpage (monoproduit) est la suivante :

$S = \{S_1, S_2, S_3, S_4\}$,

$S_1 = \{s_{a1}, s_{b1}, s_{c1}, s_{d1}, s_{d3}\}$, $S_2 = \{s_{a2}, s_{b2}, s_{c2}\}$, $S_3 = \{s_{e1}, s_{f2}, s_{g2}\}$, $S_4 = \{s_{e3}, s_{f3}, s_{g3}\}$

Cette solution conduit à la solution suivante du problème original de découpage multiproduit :

$S' = \{S'_1, S'_2, S'_3, S'_4\}$,

$S'_1 = \{a,b,c,d\}$, $p(a,1,1) = 5, p(b,1,1) = 5, p(c,1,1) = 5, p(d,1,1) = 5, p(d,3,1) = 4$

$S'_2 = \{a,b,c\}$, $p(a,2,2) = 8, p(b,2,2) = 8, p(c,2,2) = 8$

$S'_3 = \{e,f,g\}$, $p(e,1,3) = 5, p(f,2,3) = 8, p(g,2,3) = 8$

$S'_4 = \{e,f,g\}$, $p(e,3,4) = 4, p(f,3,4) = 4, p(g,3,4) = 4$

Les autres valeurs $\{p(x,k,j)\}_{x \in S,\ k=1,2,3,\ j=1,2,3,4}$ bien définies sont nulles. Le découpage en postes multiproduit est présenté dans la figure IV-9.

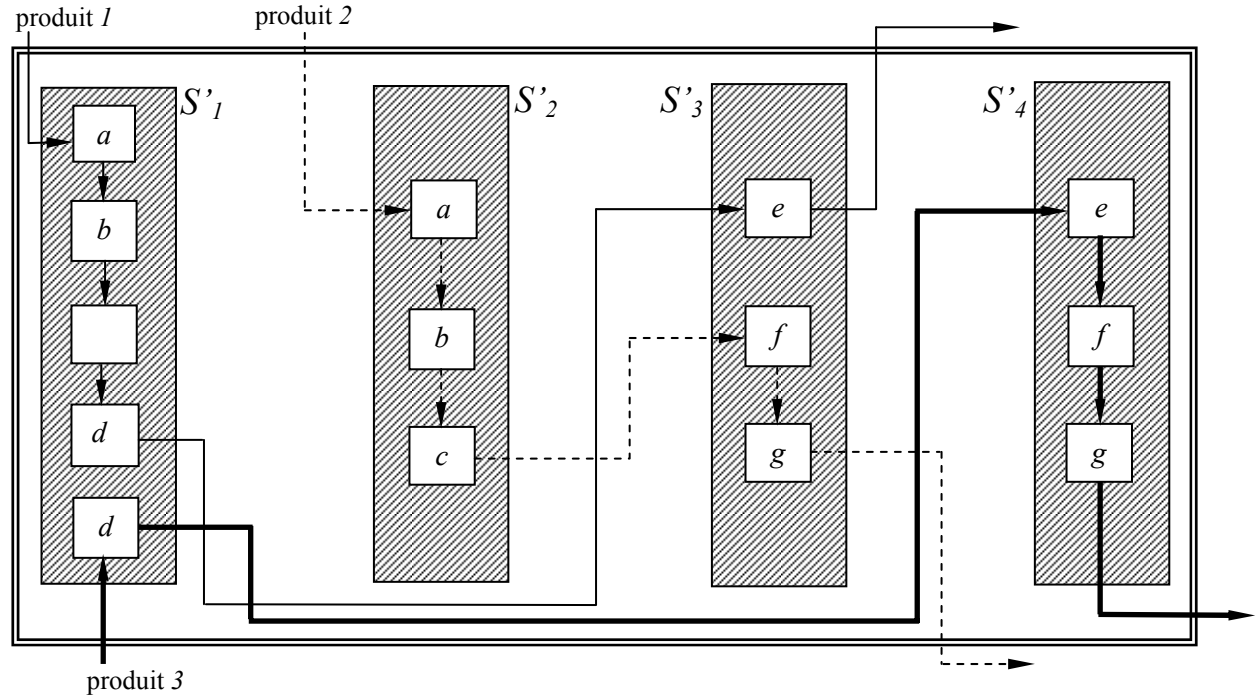


Fig. IV-9. Le découpage en postes multiproduit pour la première variante

Remarquons que la solution choisie pour le problème dérivé, de découpage monoproduit, peut être même optimale du point de vue de l'équilibrage, c'est à dire avec le nombre de postes de travail minimal.

Variante 2 *: agrégation totale des durées*

Dans ce cas, le temps de cycle est $T = 45$ unités de temps.

Nous obtenons d'abord le graphe de précédence G = (S,U) de toute la famille de produits, en tant que *l'union* des trois graphes de précédence dessinés dans la figure IV-5. Ce graphe, présenté à la figure IV-10, sera utilisé comme donnée d'entrée du problème de découpage *monoproduit*, auquel nous voulons réduire notre problème original, *multiproduit*.

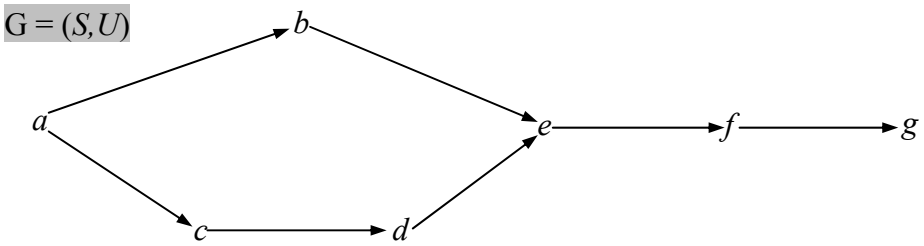


Fig. IV-10. Le graphe de la famille de trois produits pour la deuxième variante

Nous calculons les temps agrégés $\{t''(x)\}_{x\in S}$ selon la relation (IV.25) :

$t''(a)$=18, $t''(b)$=21, $t''(c)$=5, $t''(d)$=9, $t''(e)$=27, $t''(f)$=10, $t''(g)$=6

Une solution de ce problème de découpage monoproduit peut être la suivante :

$S = \{S_1, S_2, S_3\}$,

$S_1 = \{a,b\}$, $S_2 = \{c,d,e\}$, $S_3 = \{f,g\}$

La solution ci-dessus conduit à la solution du problème original multiproduit :

$S_1 = \{a,b\}$, $p(a,1,1) = 5, p(a,2,1) = 8, p(b,1,1) = 5, p(b,2,1) = 8$

$S_2 = \{c,d,e\}$, $p(c,1,2) = 5, p(c,2,2) = 8, p(d,1,2) = 5, p(d,3,2) = 4, p(e,1,2) = 5, p(e,3,2) = 4$

$S_3 = \{f,g\}$, $p(f,2,3) = 8, p(f,3,3) = 4, p(g,2,3) = 8, p(g,3,3) = 4$

Comme suite, la ligne d'assemblage multiproduit aura la structure donnée dans la figure IV-11.

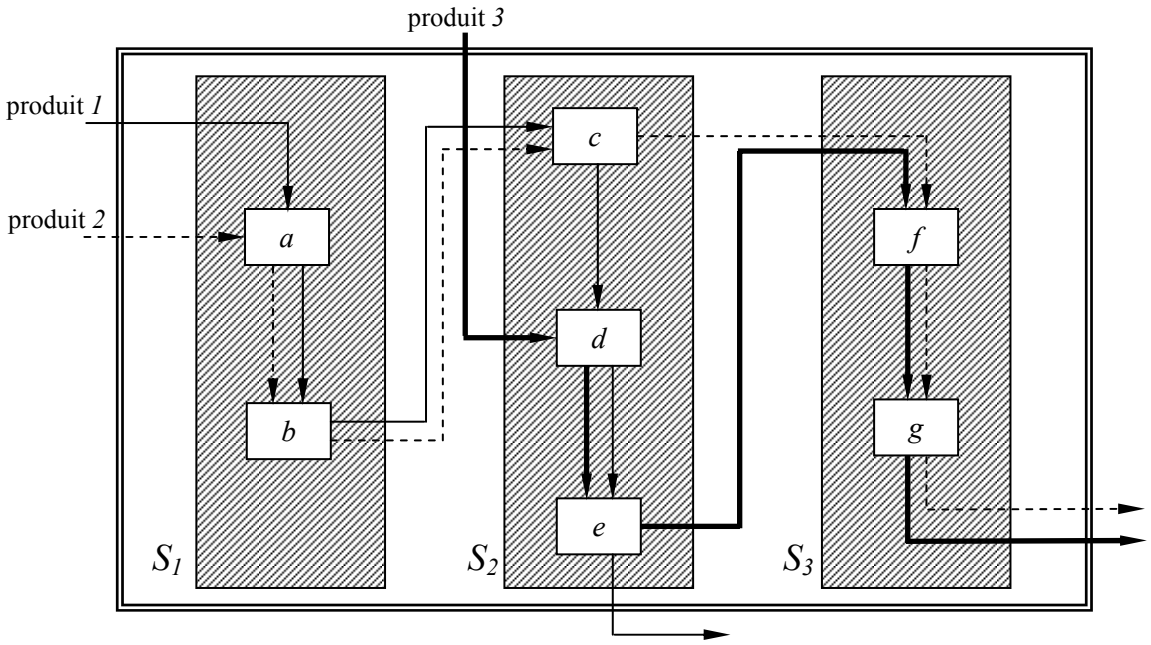


Fig. IV-11. Le découpage en postes multiproduit pour la deuxième variante

Remarque :

Les deux variantes illustrées par le dernier exemple comportent une comparaison du point de vue de leur *optimalité*. Pratiquement, cet aspect regarde la consommation de temps et de ressources matérielles. Nous pouvons dire que les deux variantes sont duales en ce qui concerne l'optimisation de ces deux catégories de consommation.

La deuxième variante conduit à une solution plus "économique" en ce qui regarde les espaces de stockage ou le nombre d'équipements et d'opérateurs, mais qui est moins bonne du point de vue de l'équilibrage, lorsque les temps non utilisés sont plus importants.

Au contraire, la première variante fournit une solution plus coûteuse, mais avec un meilleur équilibrage.

IV.2.3 Formulation systémique du problème d'équilibrage et résolution par la programmation dynamique

La formulation systémique du problème d'équilibrage des lignes d'assemblage se construit à partir d'une observation très intéressante : lorsque l'équilibrage est vu comme affectation optimale d'un nombre donné de tâches à un nombre donné de postes de travail, il peut être modélisé en tant que ***processus de prise de décision séquentiel***, qui, de plus, remplit ***un critère d'optimum***. Ce qui signifie que pour chaque poste et pour chacune des tâches il

faut décider si la tâche sera ou non assignée au poste, afin que la configuration finale du système ait le temps de cycle minimum.

Remarquons ensuite la spécificité de ce processus de prise de décision : il est *séquentiel*, il se déroule par étapes. Ce qui le rend modélisable comme ***processus dynamique discret***. Par l'intermédiaire de cette démarche, le problème de trouver l'affectation optimale est transformé dans un ***problème d'optimisation discrète***. L'avantage émerge de l'existence de méthodes efficaces de résolution, dont la plus connue est ***la programmation dynamique***.

Reprenons la formulation du problème d'équilibrage. Par rapport à sa formulation originale (voir IV.2.1), cette fois-ci nous supposons connu le nombre de postes de travail.

On donne :

- N tâches d'assemblage : $S = \{s_1, s_2, \ldots s_N\}$;
- *le graphe de précédence*, qui modélise les contraintes de précédence ;
- M postes de travail (nommés aussi machines) : $m_1, m_2, \ldots m_M$;
- τ_{ij} – le temps opératoire de la tâche t_j sur la machine m_i, $j=1,2,\ldots N$, $i=1,2,\ldots M$.

On demande :

- $\{S_i\}_{i=1,2,\ldots M}$ – partition de S respectant les contraintes de précédence ($P_i \Rightarrow S_i$, $i=1,2,\ldots M$) ;
- critère d'optimum : *minimiser le temps de cycle du système* :

$$\min \max_i \left\{ \sum_{k \in S_i} \tau_{ik} | i = \overline{1,N} \right\} \qquad \text{(IV.27)}$$

Remarque : L'expression de calcul du temps de cycle montre l'hypothèse implicite d'un porteur unique dans tout poste de travail (exécution séquentielle des tâches).

Par renumérotation, la partition $\{S_i\}_{i=1,2,\ldots M}$ sera ensuite notée $\{S_i\}_{i=0,1,\ldots M-1}$. L'affectation des N tâches sur M postes, concrétisée dans la partition $\{S_i\}_{i=0,2,\ldots M-1}$, est équivalente au **processus de prise de décision en M coups** :

$$u(i),\ i=0,1,\ldots M-1, \qquad u(i) = \begin{bmatrix} u^1(i) \\ u^2(i) \\ \ldots \\ u^N(i) \end{bmatrix}, \qquad u^k(i) = \begin{cases} 1, si\ k \in S_i \\ 0, si\ k \notin S_i \end{cases}.$$

La fonction $u(i)$ s'appelle *la fonction caractéristique* de l'ensemble S_i.

La prise de décision se fait au cours de M itérations, à partir de l'*état initial*, où aucune tâche n'est pas affectée, jusqu'à l'*état final*, où toutes les tâches sont affectées. De cette façon, un état intermédiaire quelconque aura également la signification d'une fonction caractéristique. Notons les états initial et final respectivement par $x(0)$ et $x(M)$. Ils sont décrits par :

$$x(0)=\begin{bmatrix}0\\0\\...\\0\end{bmatrix}, \qquad x(M)=\begin{bmatrix}1\\1\\...\\1\end{bmatrix}.$$

Au pas *i* le système évolue de l'état *x(i)* à l'état *x(i+1)*, suite à la décision décrite par *u(i)*. Le diagramme des transitions d'un état au suivant est analogue à la trajectoire dans l'espace d'états d'un *système dynamique discret*. Nous illustrons la trajectoire d'état dans la figure IV-12.

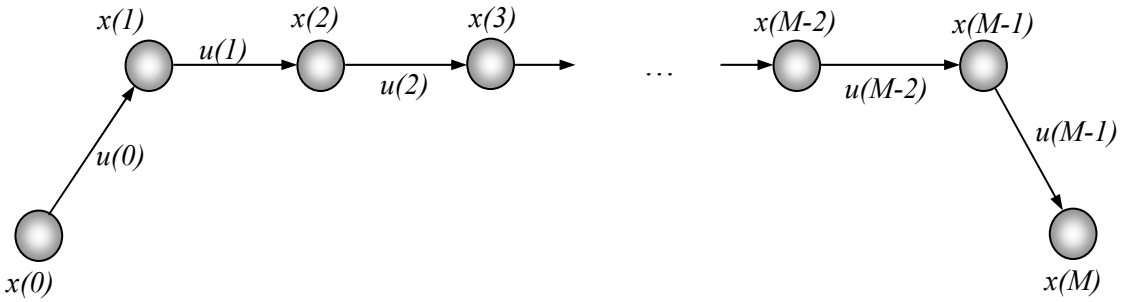


Fig. IV-12. Modélisation du processus de prise de décision séquentiel comme processus dynamique discret

Les équations qui décrivent le processus dynamique discret peuvent être aisément déduites :

$$\text{(IV.28)} \quad \begin{cases} x(i+1)=\mathrm{A}\cdot x(i)+\mathrm{B}\cdot u(i) \\ \mathrm{A}=\mathrm{B}=\mathrm{I_N}, \\ x(0)=\begin{bmatrix}0\\0\\...\\0\end{bmatrix}, \quad \text{contrainte : } x(M)=\begin{bmatrix}1\\1\\...\\1\end{bmatrix} \end{cases}$$

La programmation dynamique est une méthode d'optimisation basée sur *le principe d'optimalité* de Bellman [BELE 85]. Afin de formuler le problème d'équilibrage en termes de *programmation dynamique discrète*, il est nécessaire de transformer le critère d'optimum, exprimé par la relation (IV.27), sous la forme de l'équation de Bellman.

Pour chaque poste P_i, *i=1,2,...M*, nous définissons le vecteur des temps opératoires des tâches :

$$\tau_i^T = [\,\tau_{i1}\ \ \tau_{i2}\ \ ...\ \ \tau_{iN}\,].$$

Compte tenu de cette notation, *le temps de cycle du poste* P_i s'écrit sous la forme :

$$\sum_{j=1}^{N}\tau_{ij}\cdot u^j(i)=\tau_i^T\cdot u(i).$$

La notation :

$$\underline{u(k)}=[u(k)\ \ u(k+1)\ \ ...\ \ u(M-1)]$$

désigne le vecteur des décisions (commandes) à partir du pas *k* jusqu'à la fin.

Nous définissons ensuite *une fonction-coût*, qui doit être optimisée – en ce cas, *minimisée*. Nous utilisons la notation consacrée dans la programmation dynamique :

$$(IV.29) \qquad J(x(k),\underline{u(k)}) = \max_{i=k,k+1,\ldots M-1} \left\{\tau_i^T \cdot u(i)\right\}$$

Notons ensuite :

$$(IV.30) \qquad J^*(x(k)) = \min_{\underline{u(k)}} J(x(k),\underline{u(k)})$$

Compte tenu de la relation (IV.29), la relation (IV.30) devient :

$$(IV.31) \qquad J^*(x(k)) = \min_{u\in U_k} \max(\tau_k^T \cdot u, J^*(x(k+1))),$$

où U_k est l'ensemble des *commandes admissibles* au pas *k*.

Nous résumons ci-dessous la formulation du problème d'équilibrage comme ***problème d'optimisation discrète*** :

> Soit le système dynamique discret décrit par l'équation (IV.28) et les vecteurs τ_i^T, $i=1,2,\ldots N$.
>
> Il faut trouver la suite $\underline{u(0)} = [u(0)\ \ u(1)\ \ \ldots\ \ u(M-1)]$ telle que :
>
> 1) le système évolue entre l'état initial *x(0)* et l'état final *x(M)* imposés ;
> 2) l'évolution se fasse en minimisant $J(x(0), \underline{u(0)})$, où *J* représente la fonction-coût définie par la relation (IV.29).

L'algorithme de la programmation dynamique est basé sur *la récurrence arrière*.

Ainsi, l'étape *M* sera la première. L'algorithme reçoit les données initiales suivantes : l'état final, *x(M)*, le vecteur de commande vide, *[ ]*, et la valeur nulle de la fonction-coût, *J*(x(M))=0*.

Étape *k*

$x^1(k)$ $u^{11}(k), u^{12}(k), \ldots$ $J^*(x^1(k))$
$x^1(k)$ $u^{11}(k), u^{12}(k), \ldots$ $J^*(x^1(k))$
$x^1(k)$ $u^{11}(k), u^{12}(k), \ldots$ $J^*(x^1(k))$ ……..

Fig. IV-13. Les données mémorisées par l'algorithme de la programmation dynamique

Il en résulte que les étapes suivantes ont comme but la génération des trajectoires admissibles dont les segments finaux sont optimaux. À chaque étape *k*, *k=M, M-1, ... 1, 0*, on mémorise les *états accessibles*, accompagnés par les *commandes optimales* et les *coûts minimums*. Dans la figure IV-13 nous donnons une image suggestive des données mémorisées à l'étape *k*.

La recherche de la commande optimale impose à chaque pas *k la diminution* du cardinal de l'ensemble U_k des commandes admissibles. Cela constitue l'idée de base de l'algorithme.

Notations :

F_k – l'ensemble des tâches exécutables par le poste P_k

T_k – l'ensemble des tâches exécutables uniquement par le poste P_k

$\Omega = \mathrm{P}(S)$ – l'ensemble des sous-ensembles de S (l'ensemble de tâches)

Compte tenu des notations ci-dessus, il s'ensuit que la notation

$$\Omega_k = \{W |\ W \in \Omega,\ T_k \subset W \subset F_k\}$$

désigne l'ensemble des tâches qui peuvent éventuellement être assignées au poste P_k.

L'algorithme de la programmation dynamique se déroule sous deux types de **contraintes** :

- les contraintes *générales* ;
- les contraintes *imposées par le modèle du processus*.

Afin de détailler ces deux catégories de contraintes, nous allons expliquer d'abord ce que, au premier coup d'œil, pourrait être considéré comme un abus de notation.

Nous allons utiliser les opérateurs dédiés généralement aux opérations entre ensembles pour désigner des opérations entre vecteurs. Il s'agit particulièrement de vecteurs d'état, *x(k)*, et de commande, *u(k)*. Rappelons que le vecteur de commande, *u(k)*, représente en fait la fonction caractéristique de l'ensemble des tâches affectées au poste P_k. Les éléments du vecteur *u(k)* appartiennent à l'ensemble *{0,1}*. Donc, en un sens élargi, nous pouvons dire que *u(k)* est *un ensemble* (de tâches). Ce qui est valable également pour *x(k)*, compte tenu de la relation (IV.28). De cette manière, nous avons justifié quelques notations faites ci-dessous.

Soit *k* une étape de l'algorithme. Il existe les *contraintes générales* suivantes :

a) $u(k) \cap x(k) = \varnothing \quad \Leftrightarrow \quad (\forall j = \overline{1,N} : u^j(k) = 1 \Rightarrow x^j(k) = 0)$;

b) $u(k) \subset x(k+1) \quad \Leftrightarrow \quad (\forall j = \overline{1,N} : u^j(k) = 1 \Rightarrow x^j(k+1) = 1)$;

c) $S_k \subset F_k$;

d) $S_k \supset T_k$;

La contrainte a) exprime le fait que par la commande *u(k)* on ne peut affecter au poste P_k que des tâches qui ne sont pas encore affectées.

La contrainte b) émerge de la récurrence arrière (*x(k)=x(k+1)-u(k)*) et montre que la décision d'affecter une tâche au cours d'une étape de l'algorithme se reflète dans l'état suivant du système. Le calcul de l'état courant, *k*, se fait à partir de l'état suivant, *k+1*, comme suite à la récurrence arrière. (*Remarque* : les termes "courant" et "suivant" regardent l'évolution temporelle du système, pas le déroulement de l'algorithme.)

Les contraintes c) et d) montrent que S_k, l'ensemble de tâches caractérisé par la commande *u(k)*, doit nécessairement contenir les tâches exécutables uniquement par P_k, mais pas forcément toutes les tâches potentiellement exécutables par P_k.

Les contraintes imposées par le modèle du processus sont :

e) *u(k)* et *x(k)* soient connexes ;

f) $u(k) \in \Omega_k$.

Nous savons que la programmation dynamique est une des plus importantes méthodes d'optimisation utilisées pour résoudre le problème d'équilibrage des lignes d'assemblage. Dans ce sous-paragraphe, nous avons montré que la formulation systémique du problème d'équilibrage permet une utilisation élégante de cette méthode.

Dans le paragraphe suivant, nous allons présenter les autres méthodes d'optimisation.

IV.3 Synthèse des méthodes d'optimisation pour équilibrage

IV.3.1 Généralités

Rappelons que le deuxième aspect du problème d'équilibrage des lignes d'assemblage est le choix du "meilleur" découpage en postes. Ce qui est un problème d'optimisation sur l'ensemble des solutions de tous les découpages obtenus dans l'étape d'affectation des tâches aux postes de travail.

La différence principale entre le problème de découpage et le problème d'équilibrage repose sur *le nombre de postes de travail*. Pour le dernier problème, ce nombre est inconnu a priori et on désire qu'il soit minimisé, parce que, si le temps de cycle est minimisé, alors le nombre de postes de travail est implicitement minimisé [GUTJ 64]. Dans [GHOS 89] on trouve une bonne synthèse des méthodes d'optimisation utilisées dans le problème d'équilibrage, aussi bien dans le cas monoproduit – pour lequel l'équivalent anglais est "*Single-model Assembly Line Balancing* (**SMALB)**" – que dans le cas multiproduit ("*Mixed-model Assembly Line Balancing* – **MMALB**)".

Ce sous-paragraphe est dédié aux différentes méthodes d'optimisation utilisées pour l'équilibrage des lignes d'assemblage. Les plus importantes méthodes "***classiques***" sont : la programmation entière, la programmation dynamique, le chemin minimal dans un réseau pondéré, l'algorithme "branch-and-bound" (division-élagage) [VANA 78], [PINT 81], etc. Ces approches sont ***déterministes***. D'autre part on trouve les ***méthodes stochastiques***, qui constituent une direction assez nouvelle de recherche. Par exemple, les *techniques de descente stochastique* ont été déjà utilisée pour la résolution du problème d'équilibrage : l'algorithme du recuit simulé [JUNG 97], [HONG 97], les algorithmes génétiques [FALK 96], [SURE 96], [REKI 97], l'algorithme "Kangourou" [FLEG 95], [MÎNZ 98], etc.

Nous allons constater quelques variations intervenues dans la formulation de base du problème d'équilibrage (voir IV.2.1), qui ne changent pas sa signification de principe, mais qui le rendent plus facile pour être résolu par une plus large diversité de méthodes.

IV.3.2 Un exemple de méthode "classique"

Dans [VANA 78] le problème d'équilibrage est formulé en tant qu'affectation des tâches caractérisées par des durées fixes à *un nombre minimal* de postes de travail, sans dépasser le temps de cycle de chacun des postes et sans violer les contraintes de précédence. La solution de ce problème est *l'affectation optimalement équilibrée* des tâches aux postes de travail. Elle se trouve dans l'ensemble des solutions potentielles : les affectations des tâches

aux postes de travail (les découpages en postes), qui sont *faisables* du point de vue des contraintes temporelles et des contraintes de précédence. Ces solutions s'organisent sous une forme arborescente. Les auteurs offrent une solution de type "***branch-and-bound***" de recherche de l'optimum sur l'arbre des solutions, améliorée par l'utilisation des ***techniques heuristiques*** de réduction de la combinatoire (par exemple, la génération optimale des branches de l'arbre de solutions).

Nous allons utiliser l'acronyme "B&B" pour désigner la méthode "branch-and-bound". La recherche de l'optimum se déroule à travers une procédure itérative. Chaque itération commence avec un nœud de l'arbre de solutions, qui représente l'affectation des tâches à un seul poste de travail. Ensuite, pour chaque nœud nous associons le problème d'affectation de toutes les tâches pas encore affectées aux postes pas encore occupés. Il faut vérifier si ce problème admet une solution immédiate. Sinon, la procédure se ramifie en fournissant un nombre de nœuds correspondant aux affectations faisables du poste suivant et calcule pour chacun d'eux une limite inférieure ("lower bound"). L'algorithme choisit pour l'itération suivante le nœud qui possède la moindre limite inférieure. La découverte d'une solution immédiate pour un nœud quelconque garantit l'existence d'une solution pour ses nœuds ascendants.

Les améliorations éventuelles qui peuvent être apportées à un tel algorithme auraient comme objectif la diminution du nombre d'itérations. Parfois, pour les cas dont la combinatoire est assez grande pour que la solution optimale soit garantie après un temps de calcul excessivement long, nous pouvons fixer un nombre maximum d'itérations, ce qui conduit forcément à une solution sous-optimale.

Comme nous l'avons vu précédemment, la formulation "classique" du problème d'équilibrage, le cas déterministe monoproduit, exige que *chaque tâche soit exécutée par un seul poste de travail*. Cette condition semblait inattaquable, jusqu'au moment où Freeman et Jucker ont proposé sa relaxation [FREE 67].

Plus tard, on a formulé ***le problème d'équilibrage à postes de travail dupliqués***, avec l'objectif de minimiser les coûts d'assemblage (et ceux de production en général), sur une ligne d'assemblage modifiée, qui permet l'exécution parallèle des tâches ([PINT 81]). On définit une fonction-objectif, représentant le coût du travail par jour, qui doit être minimisée.

Nous supposons un système d'assemblage formé de q postes de travail qui fonctionnent avec un temps de cycle global de c_0. Nous désirons une nouvelle configuration dans laquelle le système fonctionne avec un temps de cycle de c_1. La méthode proposée pour la restructuration est de "doubler" certains postes, afin que les exécutions des leurs tâches soient parallèles.

La fonction-coût est définie ainsi :

$$C_{t/j} = r \cdot (q + q \cdot R \cdot \Delta c / c_1),$$

où les notations signifient :

$C_{t/j}$ – le coût du travail dans un jour

r – le coût unitaire (par unité de produit) du travail

R – le rapport entre le coût supplémentaire et le coût régulier du travail

$$\Delta c = \begin{cases} c_0 - c_1 \text{ , si } c_0 > c_1 \\ 0 \text{ , sinon} \end{cases}$$

Cette approche montre l'utilisation de l'algorithme B&B adapté aux nouvelles contraintes introduites par le parallélisme des postes et amélioré à l'aide des deux stratégies de réduction de la combinatoire.

Il est important d'observer que le parallélisme implique soit des *tâches dupliquées*, soit des *postes dupliqués*, mais ces deux situations ont comme résultat l'introduction de moyens supplémentaires de production, qui demandent, donc, des coûts supplémentaires. Dupliquer les postes signifie que la ligne d'assemblage acquiert une plus grande flexibilité et un temps de cycle réduit, parce que la durée permise d'exécution à chaque poste est multipliée par *n* si le poste est multiplié par *n*.

La formulation du problème d'équilibrage à postes parallèles repose sur *la minimisation des coûts globaux de production* – aussi bien ceux de la main d'œuvre, que ceux de fabrication proprement dite – à condition d'obtenir une valeur imposée du taux de production, qui est en fait une expression équivalente du temps de cycle. Dans la littérature, ce problème est connu sous le nom de "**ALBPS**" (l'acronyme anglais dérivé de "*Assembly Line Balancing with Paralleling of Stations*"). Il est en *fait un problème d'optimisation sous contraintes* ([PINT 81]).

L'interprétation économique de la formulation du problème ALBPS conduit à la conclusion que les coûts pris en calcul ne sont pas les plus relevants, mais considérer encore d'autres coûts va entraîner l'impossibilité de résoudre même le plus simple problème d'équilibrage.

IV.3.3 Méthodes stochastiques concernant l'équilibrage

Comme nous l'avons montré, l'équilibrage des lignes est essentiellement un problème d'optimisation discrète.

Toute méthode d'optimisation discrète pose le problème de la *complexité*. Le problème d'équilibrage est NP-*complet*, lorsqu'il s'agit d'obtenir une partition optimale d'un graphe. Il en résulte qu'un algorithme qui garantisse l'obtention d'un minimum global pour ce problème aurait une complexité exponentielle. On ne peut pas implanter un tel algorithme en régime de temps réel, surtout dans des grands systèmes d'assemblage.

Dans ces conditions, les méthodes stochastiques présentent un intérêt de plus en plus élevé, puisqu'elles offrent *une solution approximative*, sous-optimale, mais qui peut être satisfaisante en pratique. Dans ce sous-paragraphe nous allons passer en revue trois méthodes stochastiques : l'algorithme du recuit simulé, les algorithmes génétiques et l'algorithme "Kangourou", un cas particulier de technique de descente stochastique.

Un problème très important de méthodes stochastiques est de garantir leur ***convergence***. Chaque classe de méthodes stochastiques possède des moyens spécifiques pour garantir la convergence. La garantie de la convergence d'un algorithme d'optimisation stochastique nous permet de définir la marge dans laquelle la solution fournie – qui est sous-optimale – peut être considérée comme optimale.

IV.3.3.1 L'algorithme du recuit simulé

Dans [HONG 97] on trouve une approche du problème d'équilibrage à partir du coût global d'assemblage. Il s'agit d'*une méthode stochastique* d'optimisation appliquée au cas déterministe : la méthode du recuit simulé, qui vient d'une analogie avec le processus d'atteinte d'équilibre thermique au sein de la structure interne d'un métal chauffé. L'équilibre thermique correspond à une structure cristalline caractérisée par un minimum de l'énergie.

Dans ce travail, l'optimalité d'un système d'assemblage est traitée *simultanément* du point de vue de deux aspects : ***la minimisation du coût global d'assemblage et la minimisation du temps d'inactivité de chaque poste***. À chaque affectation tâches-postes on associe *une fonction d'énergie*, qui donne l'expression du coût global, en caractérisant également le degré d'équilibrage.

Ainsi, le problème d'équilibrage est défini comme problème de minimisation de cette fonction, en respectant les *contraintes d'assemblage* – de *précédence* et de *connexité*. Les contraintes de précédence regardent le modèle du processus d'assemblage, c'est à dire le graphe de précédence, et les contraintes de connexité regardent le modèle du produit à assembler.

L'affectation optimale est élaborée selon une procédure itérative, dont la description générale est donnée à la figure IV-14.

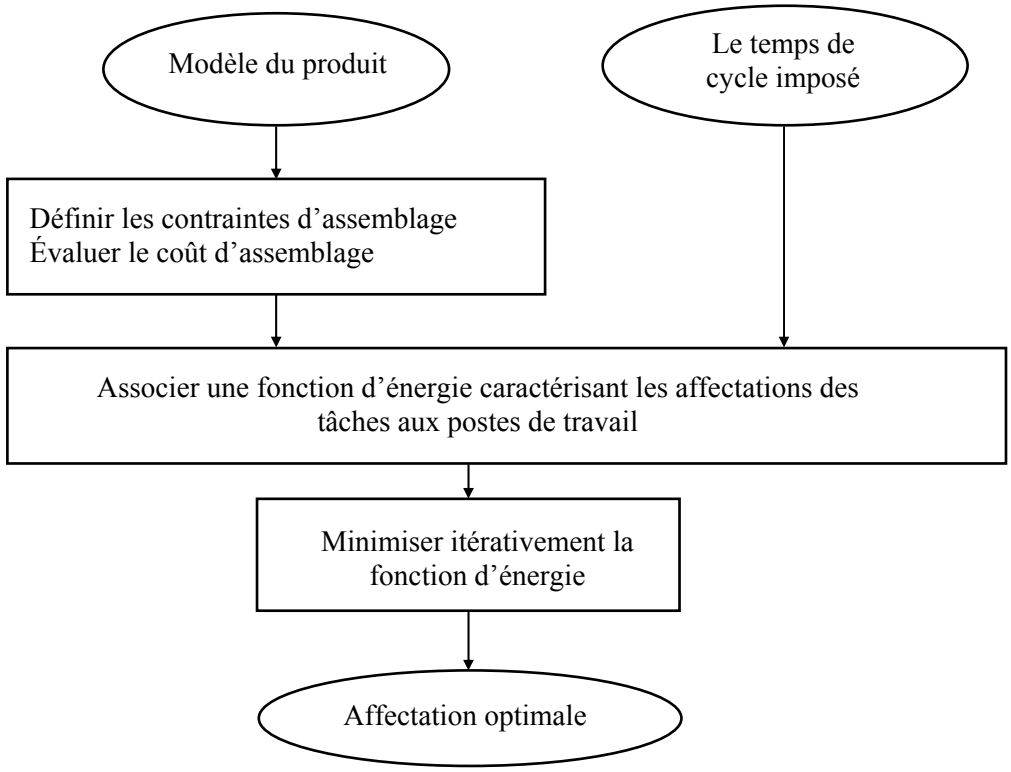


Fig. IV-14. L'algorithme du recuit simulé appliqué au problème d'équilibrage des systèmes d'assemblage

La convergence de l'algorithme du recuit simulé est garantie sous certaines conditions. Nous ferons ci-dessous quelques observations concernant la vitesse de convergence.

Le principe de l'algorithme du recuit simulé est de minimiser la fonction d'énergie selon une loi qui imite la décroissance de la température d'un métal vers l'équilibre thermique. Donc, l'algorithme d'optimisation sera conduit par une fonction de changement de la température. Un bon choix de l'expression mathématique de cette fonction assure la ***convergence asymptotique*** de l'algorithme vers le minimum global. Par exemple, la loi de *décroissance logarithmique* de la température en fonction du temps mesuré en nombre d'itérations m :

$$T(m) = \frac{T_0}{1 + \ln(m)}$$

où T_0 désigne la température initiale.

Le désavantage principal de cette loi de variation est le temps de calcul prohibitivement long, lorsque la température baisse assez lentement aux valeurs grandes de l'énergie - au début de l'algorithme - et, donc, le système se dirige très lentement vers optimum. C'est la raison pour laquelle Szu et Hartley ont proposé en 1987 le remplacement de la loi de probabilité de Boltzmann par la loi de Cauchy ([DUMI 97]). Ainsi, le paramètre T a la variation suivante :

$$T(m)=\frac{T_0}{1+m}$$

et la probabilité d'avoir une variation de la fonction d'énergie de x entre deux itérations successives est :

$$P(x)=\frac{T(m)}{T^2(m)+x^2}$$

IV.3.3.2 Les algorithmes génétiques

Plusieurs travaux de recherche proposent que les problèmes d'optimisation combinatoire soient résolus à l'aide des algorithmes génétiques. Par exemple, dans [SURE 96] on rencontre une telle approche du problème d'équilibrage, ***le cas stochastique***, où on suppose comme connu la distribution du temps de cycle de chaque poste de travail. Les algorithmes génétiques, surnommés "métaphores biologiques", sont des méthodes stochastiques de recherche qui formalisent la voie suivie par la nature pour atteindre l'optimum [REND 95]. L'idée de base est de copier le mécanisme d'adaptation d'une population hétérogène aux changements de l'environnement.

Nous pouvons dire que l'utilisation des algorithmes génétiques conduit à des formulations très particulières pour chaque problème à résoudre, ce qui requiert une connaissance profonde du problème en cause. En termes d'algorithmes génétiques, chaque problème doit être adéquatement "*codé*". Il s'agit en principe de correctement définir les notions de base manipulées par tout algorithme génétique, par exemple : *individu*, *population*, *opérateurs génétiques* – de croisement et de mutation – *fonction-objectif* (fonction-critère ou *fonction d'adéquation*) qui doit être, cette fois-ci, maximisé, puisqu'elle modélise le degré d'adaptation d'une population à l'environnement.

Dans [SURE 96] le but du problème d'équilibrage est de trouver l'affectation des tâches, telle que :

- le temps de cycle du système, T_c, ne soit pas dépassé ;
- les contraintes de précédence soient respectées ;
- l'indice d'équilibrage (ou de "lissage") du système soit minimisé :

  $$Minimiser\ SI=\sqrt{\sum_{k=1}^{n}(S_{max}-S_k)^2}$$, où :

  n - le nombre de postes de travail du système ;
  S_{max} - le temps de cycle maximum sur l'ensemble de postes ;
  S_k - le temps de cycle du poste k ;
- la probabilité d'arrêt de la ligne d'assemblage soit minimisée :

  $$Minimiser\ P = 1-\prod_{i=1}^{n} P_{s_i}$$,

  où P_{s_i} est la probabilité que le poste i ne dépasse pas le temps de cycle du système.

Jusqu'à ce point-là, il ne s'agit que d'une formulation semblable à celles que nous avons présentées auparavant. Mais l'approche de ce problème par algorithmes génétiques comporte plusieurs détails d'implémentation, concernant sa "traduction" en termes d'algorithmes génétiques (voir le travail cité). Ce qui reste valable d'ailleurs pour tout problème particulier à résoudre par cette approche.

La convergence des algorithmes génétiques est garantie par ***le théorème du schéma***, qui se rapporte à la manière de codage par chaînes de bits [REND 95]. L'interprétation de ce théorème repose sur le comportement à la longue d'une population soumise à un certain changement de l'environnement : les individus, fortement différents au début, forment finalement des groupes optimalement adaptés. En général, ces groupes correspondent aux points de maximum locaux.

Dans la mesure où il prouve la convergence des algorithmes génétiques, ce théorème a une grande importance pratique, mais il n'explique que le début de l'évolution. Grâce à la garantie de la convergence, on peut surmonter le principal désavantage des algorithmes génétiques, *le temps de calcul assez long*, en définissant le point pour lequel on peut considérer une solution sous-optimale comme solution optimale.

IV.3.3.3 Techniques de descente stochastique – l'algorithme "Kangourou"

L'algorithme "Kangourou" est une implantation spéciale d'une technique de descente stochastique [FLEG 95].

Dans [MÎNZ 98] on présente l'utilisation de cet algorithme pour résoudre le problème d'équilibrage, aussi bien dans le cas *monoproduit*, que dans le cas *multiproduit*. Nous allons détailler deux aspects importants au sujet de cette approche :

- le modèle du processus d'assemblage utilisé ;
- le principe de l'algorithme.

Cette approche repose sur l'utilisation du **graphe de précédence** – noté par G=(*S,U*) – comme modèle du processus d'assemblage.

La signification de ce modèle est très claire dans le cas monoproduit (voir le troisième chapitre de ce travail). Afin d'obtenir un traitement unitaire, par le même algorithme, des deux cas approchés, nous adoptons ***deux hypothèses*** assez réalistes :

- l'hypothèse d'une *famille iso-structurelle* de produits ;
- l'hypothèse de l'*agrégation totale des durées* des tâches.

L'hypothèse d'une famille iso-structurelle nous permet de *généraliser* la notion de "graphe de précédence". C'est à dire, si tous les produits de la famille sont identiques du point de vue structurel, étant formés de *parties génériques*, alors il est possible de construire un graphe de précédence unique pour toute la famille.

De plus, nous supposons que, pour tous les produits, une même opération d'assemblage est exécutée dans le même poste de travail. Ce qui est une supposition réaliste, lorsque les postes possèdent en général des opérateurs spécialisés. Cette hypothèse correspond à l'agrégation totale des durées des tâches. Ainsi, la manière dont nous construisons un graphe de précédence représentant toute la famille de produits devient claire (voir IV.2.2).

Le principe de l'algorithme "Kangourou" est semblable à celui de l'algorithme du recuit simulé : *la minimisation d'une fonction-objectif – f(u)* – à l'aide d'une *procédure itérative* qui contient deux parties : la ***descente stochastique*** et le "***saut***". Cette fonction peut avoir même des *valeurs infinies*, ce qui n'est pas valable pour l'algorithme du recuit simulé.

La procédure de descente stochastique est basée sur une ***sélection aléatoire***. Ainsi, une solution *u* du problème – qui est une partition sur l'ensemble *S* de tâches, ou, autrement dit, un découpage en postes de travail – est remplacée par une solution meilleure, située dans le voisinage N(*u*) de la première. S'il n'est pas possible d'obtenir une nouvelle amélioration, on exécute un "saut" aléatoire, afin d'échapper à l'attraction d'un point de minimum local.

Nous soulignons que les deux parties – la descente et le saut – peuvent utiliser des *définitions différentes du voisinage*.

Notons $u = \{S_1, S_2, \ldots S_M\}$ une partition sur l'ensemble S, où M est le nombre de postes. La définition du voisinage requiert la définition de deux ***opérateurs de déplacement*** : à gauche ("**sh**ift-**l**eft"), respectivement à droite ("**sh**ift-**r**ight") :

$$(u, s_j) \xrightarrow{shl} u'$$

$$(u, s_j) \xrightarrow{shr} u'$$

L'application d'un opérateur de déplacement à une partition u par rapport à une tâche s_j produit une nouvelle partition u', où la tâche $s_j \in S_k$ est déplacée au poste S_{k-1}, $1<k\leq M$, respectivement au poste S_{k+1}, $1\leq k<M$.

Nous disons qu'un opérateur de déplacement est *bien défini* s'il produit une partition u' qui respecte les contraintes de précédence.

Dans les deux cas, si la partition u respecte les contraintes de précédence imposées par le graphe G, il existe aussi une deuxième condition qui doit être remplie pour que la partition résultante, u', respecte également les contraintes de précédence :

- déplacement à gauche : tous les prédécesseurs directs de s_j appartiennent à des postes de type S_i, $i<k$;
- déplacement à droite : tous les successeurs directs de s_j appartiennent à des postes de type S_i, $i>k$.

> Un voisinage d'une partition u – noté N(u) – est l'ensemble des partitions u' résultant de l'application d'un opérateur de déplacement qui est bien défini. La tâche qui sera déplacée est choisie conformément à une loi de distribution uniforme.

Ensuite, nous allons présenter quelques aspects regardant **la complexité** et **la convergence** de l'algorithme "Kangourou".

La procédure itérative peut commencer avec une solution quelconque, préférablement avec une solution qui correspond à un découpage réel en postes. Une bonne solution initiale s'obtient, par exemple, par SAP – Sequential Assignment Procedure [KUSI 94] – qui a comme but l'obtention de chargements aussi égaux que possible pour tous les postes.

Si la partition initiale du processus itératif respecte les contraintes de précédence, alors, à l'aide d'un opérateur de déplacement qui est bien défini, l'algorithme va générer *seulement* les partitions qui respectent également ces contraintes. Il s'ensuit que l'algorithme aura une ***complexité faible***.

Quant à la convergence de l'algorithme, le travail cité ([MÎNZ 98]) propose la démonstration par ***l'approche classique des chaînes de Markov***. La suite $\{u^*_n\}_{n=1,2,\ldots}$ des meilleures solutions, issue d'une exécution quelconque de l'algorithme, est une réalisation d'une chaîne de Markov [FLEG 95].

On montre que le processus stochastique (u^*_n) converge avec la probabilité *1* vers le point de minimum global. La démonstration s'appuie sur ***la propriété d'accessibilité*** de la fonction de voisinage.

Pour définir le voisinage dans le cas du *saut aléatoire*, nous pouvons adopter la même définition que celle utilisée par la procédure de descente stochastique : N'(u)=N(u). Dans le cas où nous intégrons des procédures heuristiques déterministes au sein de la procédure de descente, afin de guider la recherche de l'optimum, nous devons modifier la définition de N(u) : N(u)⊂N'(u).

IV.4 Conclusion

Les problèmes de l'affectation des tâches aux postes de travail et d'équilibrage appartiennent à la classe des **méthodes de conception** des systèmes d'assemblage. Une affectation des tâches aux postes s'appelle aussi découpage en postes.

Dans ce chapitre nous avons montré que ces deux problèmes sont étroitement liés : l'équilibrage est formulé comme problème d'affectation optimale. Nous avons montré que l'équilibrage des systèmes d'assemblage peut s'exprimer par plusieurs critères d'optimum, dont le plus utilisé est **la minimisation du temps de cycle total**.

Nous avons présenté la **formulation mathématique** du problème d'équilibrage, aussi bien dans le cas **monoproduit**, que dans le cas **multiproduit**.

Puisque l'équilibrage se réfère au "meilleur" découpage en postes, nous avons présenté premièrement la résolution du problème de découpage en postes, illustrée par un exemple d'algorithme de découpage en postes pour les deux cas. Lorsque nous avons montré que les composantes connexes du graphe d'indifférence ont la structure de poste de travail, elles peuvent constituer une des solutions du problème de découpage.

Le cas multiproduit peut être traité ***par réduction*** au cas monoproduit en utilisant deux variantes d'agrégation des durées des tâches : *l'agrégation partielle* et *l'agrégation totale*. Nous avons fourni un exemple pour illustrer ces deux variantes.

La formulation du problème d'équilibrage comporte comme donnée d'entrée **le graphe de précédence** en tant que modèle du processus d'assemblage. Nous connaissons la signification du graphe de précédence dans le cas monoproduit. Cette signification peut être généralisée au cas multiproduit, quand il s'agit d'**une famille iso-structurelle de produits**.

Donc, concevoir un système d'assemblage équilibré est essentiellement un **problème de partition optimale sur un graphe**. Comme nous le savons, ce type de problème est intrinsèquement **NP-complet**. Pour cette raison, nous avons passé en revue quelques méthodes d'optimisation concernant l'équilibrage, basées sur ***la recherche heuristique***.

Il existe deux grandes classes de méthodes d'optimisation pour l'équilibrage : les méthodes **déterministes** (nommées "classiques") – dont nous avons brièvement présenté la programmation dynamique et l'algorithme "branch-and-bound" – et les méthodes **stochastiques** – illustrées par l'algorithme de recuit simulé, les algorithmes génétiques et les techniques de descente stochastique.

Nous avons proposé une *formulation systémique* du problème d'équilibrage, adaptée à la résolution par programmation dynamique. Ensuite, nous avons justifié la nécessité des méthodes stochastiques par leur avantage de fournir une **solution sous-optimale**, acceptable dans une marge convenablement définie, une fois que leur **convergence** est garantie. Nous avons montré que la démonstration de la convergence repose en principe sur la *convergence en probabilité* des processus stochastiques.

L'approfondissement du problème d'équilibrage nous a aidé à souligner les **aspects fondamentaux** de l'approche "classique", au sens de son utilisation pour la plupart des systèmes d'assemblage.

En connaissant les notions fondamentales et les méthodes généralement utilisées pour résoudre ce problème, nous allons présenter dans le chapitre suivant une étude d'une classe spéciale de systèmes d'assemblage, qui ont la capacité d'**auto-organisation**. Ce qui signifie que leurs configurations changent dans le temps, en évoluant *spontanément* vers *la meilleure configuration au sens de l'équilibrage*. De tels systèmes sont dits **systèmes d'assemblage avec auto-équilibrage**.

V SYSTEMES D'ASSEMBLAGE AVEC AUTO-EQUILIBRAGE

V.1 Introduction. Objectif

Ce chapitre est consacré à l'étude d'un cas particulier de systèmes d'assemblage, qui possèdent une propriété remarquable : **l'équilibrage spontané,** ou **l'auto-équilibrage**. Cette propriété intrinsèque résulte comme conséquence d'un principe particulier d'organisation, qui est également applicable à un système de production quelconque.

Lors de ce que nous avons présenté dans le chapitre antérieur, un système d'assemblage équilibré est un système dont le découpage en postes assure un temps de cycle total minimum. Il faut remarquer qu'une méthode de conception soumise à un tel critère d'optimum a comme résultat *une structure bien définie* du système, qui fonctionne de manière optimale dès le début et qui ne devra souffrir aucun changement pendant l'exploitation, sauf s'il survient une panne, ce qui requiert en général une réaffectation des tâches aux postes.

Au contraire, la classe de systèmes d'assemblage étudiée dans ce chapitre n'est pas caractérisée par un temps de cycle constant pendant toute la période de fonctionnement. Ces systèmes, sous certaines conditions, atteignent ***spontanément, sans intervention consciente***, un régime permanent, caractérisé par un taux de production constant – ou, autrement dit, ***un temps de cycle*** *constant* – qui, de plus, est ***optimal***. Un tel système s'appelle système d'assemblage avec auto-équilibrage. Dans la littérature ces systèmes sont désignés par le terme anglais de "*bucket brigades*".

Au sein des systèmes d'assemblage avec auto-équilibrage il n'existe plus une affectation fixe des tâches aux postes de travail, puisque la spécificité de ces systèmes est l'utilisation d'***opérateurs humains***, qui *peuvent bouger* entre postes adjacents pour continuer leur travail sur un produit. Les ouvriers travaillent en respectant certaines règles qui sont intégrées dans *une stratégie de contrôle décentralisée*.

Nous pouvons dire qu'en ce cas, les notions de "tâche d'assemblage" et de "poste de travail" se confondent : sur chaque poste on exécute une seule tâche. Dans ce contexte, la notion de "*poste*", comprise comme ensemble de tâches, peut être attribuée à l'ensemble de postes successifs sur lesquels un ouvrier quelconque travaille au cours d'un cycle de fabrication. Donc, *un poste* "classiquement" défini est en fait *formé par plusieurs postes*.

Ainsi, la notion de "découpage en postes" n'a plus la signification de partition fixe sur l'ensemble des tâches. Lorsqu'on permet aux ouvriers de bouger d'un poste au suivant sur la

ligne, nous pouvons parler d'un ***découpage dynamique***, avec des frontières variables dans le temps. L'approche systémique de tels systèmes a permis de montrer *une condition suffisante* pour que le découpage dynamique évolue vers une forme fixe, qui reste inchangée à partir d'un certain moment et qui correspond au temps de cycle minimum pour le système donné.

L'évolution spontanée vers un *comportement stabilisé* – qui est *périodique de durée égale au temps de cycle minimum* – présente un grand avantage du point de vue de la conception et aussi du contrôle du système.

Au cours de ce chapitre nous nous intéressons à l'**analyse** des lignes d'assemblage auto-équilibrées, qui repose sur les méthodes d'analyse des systèmes dynamiques, en ignorant les aspects particuliers concernés par l'assemblage.

Tout d'abord, nous allons présenter en détail **les hypothèses de fonctionnement** de ce type de lignes et **les approches de modélisation** rencontrées dans la littérature, qui ont conduit à quelques résultats théoriques importants. Une **analyse par simulation** des différents cas de figures sera présentée pour illustrer le comportement de cette classe de systèmes d'assemblage.

Ensuite, nous allons proposer un nouveaux point de vue, en montrant l'opportunité de regarder les lignes avec auto-équilibrage en tant que **systèmes dynamiques hybrides**. Nous allons encadrer ce type de lignes dans un formalisme qui va nous permettre l'étude de la stabilité et **une nouvelle démonstration** de l'existence du comportement périodique optimal.

V.2 Le principe de l'auto-équilibrage dans les systèmes d'assemblage

V.2.1 Description et fonctionnement d'une ligne de type "bucket brigade"

L'idée de "bucket brigade" a connu sa première implantation pratique dans l'industrie textile, sous la direction de Toyota, dans les années '70, sous le nom de "*Toyota Sewn Management System*" – brièvement, TSS (marque de commerce de Aisin Seiki Co., Ltd). Aux États Unis la production de type "bucket brigade" a été premièrement implantée chez Riverside Fashions of Norris, en 1989. Ultérieurement, le nombre des entreprises qui ont organisé le flux de production sur cette idée s'est bien agrandi.

> Dans ce qui suit, pour des raisons de simplicité, nous allons utiliser le terme de "**ligne (de type) TSS**" pour désigner une ligne de type "bucket brigade".

L'étude théorique de ce type de ligne de fabrication a été commencée en 1993, par deux auteurs américains, J.J. Bartholdi et D.D. Eisenstein [BART 95], [BART 96a]. L'approche développée par les deux auteurs utilise quelques hypothèses simplificatrices, en s'appuyant sur la théorie des *systèmes dynamiques stochastiques*. Les premiers résultats valorisants ont été obtenus en modélisant l'évolution d'une ligne auto-équilibrée à l'aide d'*une chaîne de Markov*. Les mêmes auteurs ont continué leur démarche, en ajoutant de nouveaux éléments au modèle initial [BART 96b], [BART 98], [BART 99a], [BART 99b].

Une ligne de fabrication de type TSS peut être imaginée dans tous les types d'industrie, partout où les produits passent d'un ouvrier à un autre. L'exemple d'une ligne d'assemblage organisée de telle façon est suggestif pour tous les domaines concernés par la fabrication. Sur une telle ligne, les produits subissent des opérations successives d'assemblage et avancent vers le but de la ligne, où ils sortent comme produits finis.

La difficulté de concevoir une ligne équilibrée et, donc, avec un taux de production maximum, est bien diminuée pour les lignes de type TSS, à cause de la propriété d'***auto-organisation***. Dans une certaine mesure, l'idée est semblable à celle d'organisation des insectes "sociales", comme les abeilles ou les fourmis, où la coordination globale émerge spontanément, comme suite des interactions multiples entre plusieurs composantes simples. Il en est de même dans les lignes TSS, où on impose que chaque ouvrier suive *la même règle* qui lui dit ce qu'il doit faire ensuite. Cette règle sera appelée "**la règle TSS**", selon sa première utilisation.

Les éléments de base de l'organisation des lignes TSS sont présentés ci-dessous. Nous utilisons les **notations** suivantes :

n - le nombre des ouvriers

m - le nombre des postes de travail

Sur une ligne TSS le nombre d'ouvriers est strictement inférieur à celui des postes : $n<m$. Une instance du produit à assembler sera appelée *article*. Les ouvriers bougent entre postes adjacents pour continuer le travail sur un article. Il ne peut exister qu'*au plus un ouvrier sur un certain poste* à un instant donné.

La règle TSS contient deux parties :

CHAQUE OUVRIER BOUGE À SA VITESSE DE TRAVAIL	(**déplacement en avant**) Continuer le travail sur un seul article, puis le passer sur des postes successifs (où pour chaque poste *l'ouvrier d'indice supérieur a priorité*). Si l'article est pris par le successeur (ou, pour le dernier, quand il finit le travail sur l'article), alors abandonner l'article et commencer le déplacement en arrière.
COMME SUITE DE LA REINITIALISATION – qui se produit quand le dernier ouvrier finit son travail CHAQUE OUVRIER BOUGE A VITESSE INFINIE – *DUREE 0*	(**déplacement en arrière)** Déplacement en arrière et récupération de l'article du prédécesseur (ou, pour le premier, prendre des matières premières pour commencer un nouvel article). Commencer le déplacement en avant.

Commentaires au sujet de la règle TSS

Au cours de la **première partie** de la règle, chaque ouvrier passe l'article sur des postes successifs, en travaillant à sa ***vitesse de travail***. La signification de la vitesse de travail sera détaillée plus loin, quand nous présenterons les hypothèses de modélisation des lignes TSS.

Remarquons que les ouvriers sont obligés de *maintenir leur ordre* sur la ligne, puisque l'ouvrier d'indice supérieur a toujours priorité. Donc, un ouvrier d'indice *i* peut éventuellement être *bloqué* quand il devrait passer sur un poste occupé par l'ouvrier d'indice $i+1$, $i=1,2,...n-1$. Le dernier ouvrier, d'indice *n*, n'est *jamais bloqué*.

Cette situation se maintient jusqu'au moment où l'article est pris par le successeur et l'ouvrier doit l'abandonner et commencer à se déplacer en arrière.

Le **déplacement en arrière** a lieu *dans le même instant* pour tous les ouvriers, plus précisément, lorsque le dernier finit son travail. Un tel instant s'appelle ***réinitialisation*** de la ligne. Il est raisonnable de considérer que chaque ouvrier se déplace en arrière à *vitesse pratiquement infinie* par rapport aux vitesses de travail. Il en résulte que la réinitialisation de la ligne se fait ***instantanément*** et puis les ouvriers commencent le déplacement en avant.

En résumé, nous formulons une **conclusion** importante :

> Le fonctionnement d'une ligne TSS est une suite de déplacements en avant, à partir des positions initiales dictées par les instants de réinitialisation, quand chaque ouvrier abandonne sa position et recommence de la position abandonnée par son prédécesseur.
> Le premier ouvrier recommence toujours à partir de *0*.

Nous allons voir ensuite comment le fonctionnement décrit ci-dessus peut être formalisé mathématiquement.

V.2.2 Modélisation des lignes TSS – notations et hypothèses [BART 96a]

Nous allons présenter ce qui peut être considéré comme *modèle de base* des lignes TSS. Sa simplicité nous aidera à surprendre l'essentiel du principe d'organisation de telles lignes de fabrication. Ce modèle initial peut être ultérieurement enrichi, en permettant une étude plus approfondie.

Tout d'abord, nous introduisons et expliquons quelques **notations** spécifiques.

Le contenu total standard de travail nécessité par un article sera représenté comme un segment normalisé $[0,1]$, partionné en intervalles qui correspondent aux postes de travail : à chaque poste correspond une fraction notée p_j, $j=1,2,...m$. Évidemment, il existe la relation : $\sum_{j=1}^{m} p_j = 1$. Une représentation suggestive est donnée à la figure V-1.

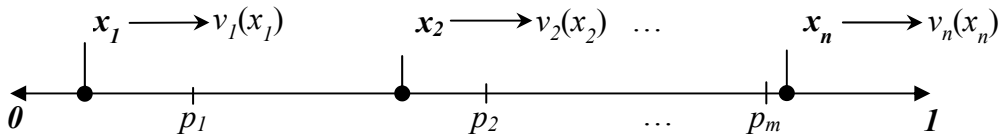


Fig. V-1. Représentation des postes de travail comme fractions du contenu total de travail

Dans cette figure nous avons adopté aussi la notation x_i pour désigner *la position instantanée* de l'ouvrier *i* sur la ligne. Cette position représente ***la fraction cumulée du travail*** effectué jusqu'à ce moment-là sur l'article.

Notons par :

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, \qquad 0 \le x_1 \le x_2 \le \dots \le x_n \le 1$$

le vecteur des positions instantanées, dont les éléments respectent, comme *conséquence de la règle TSS*, la relation d'ordre susmentionnée au sein de l'intervalle [*0,1*].

Les ouvriers sont modélisés par des ***fonctions de vitesse (de travail)***, qui dépendent de leurs positions instantanées. Dans une implantation pratique, ces vitesses varient en fonction des opérations exécutées par les ouvriers. Elles modélisent *l'habileté de travail* des ouvriers.

Donc, par la notation $v_i(x)$ nous désignons *la vitesse instantanée* de travail de l'ouvrier *i* à la position *x*, quand il n'est pas bloqué par un poste occupé. On demande que les fonctions de vitesse remplissent deux conditions :

a) qu'elles soient *continues* presque partout, sauf éventuellement au passage d'un poste au suivant (le cas où un ouvrier est obligé de s'arrêter à l'entrée d'un poste occupé) ;

b) $\exists\, b, B: \forall\, i, i = 1,2...n, \forall\, x \in [0,1]: 0 < b < v_i(x) < B < \infty$.

La deuxième condition reflète le fait que les ouvriers ne sont ni infiniment rapides, ni infiniment lents. Donc, les valeurs des fonctions de vitesse se situent dans un intervalle fini, [*b,B*], de nombre réels positifs.

Nous considérons ensuite que les ouvriers peuvent être ordonnés du point de vue de leur habileté de travail à l'aide d'une *relation d'ordre total sur l'ensemble des fonctions de vitesse*. Notons cette relation d'ordre par "$\prec$". Elle est définie par la suite :

$$(V.1) \qquad v_i \prec v_j \Leftrightarrow sup_{x\in[0;1]}\left(\frac{v_i(x)}{v_j(x)}\right) < 1$$

À cet écriture nous donnons l'interprétation suivante : l'ouvrier *j* est ***plus rapide*** que l'ouvrier *i* s'il travaille plus rapidement que *i* pour toute opération de la ligne. Par conséquent, pour toute ligne TSS, il existe *le plus lent ouvrier* et *le plus rapide* ouvrier.

Comme conclusion, nous pouvons résumer **les hypothèses** du modèle le plus simple d'une ligne TSS (connu également sous le nom de "*modèle normatif*") :

1) ***ordonnancement total des ouvriers selon leurs vitesses de travail*** – chaque ouvrier est caractérisé par une fonction de vitesse distincte ;

2) ***durée insignifiante du déplacement en arrière*** – lorsque les ouvriers bougent en arrière à une vitesse infiniment plus grande que leurs vitesses de travail ;

3) ***"lissage" et prédictibilité du travail*** – le contenu standard du travail nécessité par un article, qui est normalisé à *1*, est continûment et uniformément distribué à travers la ligne.

Notons par

$$\left\{\underline{x}^{(0)}, \underline{x}^{(1)}, \underline{x}^{(2)}, ...\underline{x}^{(t)}, ...\right\}$$

la séquence des positions ***immédiatement après*** les réinitialisations successives ($x_1^{(t)} = 0, x_n^{(t)} = 1, \forall t \in \mathbf{N}$). Cette séquence est appelée l'***orbite*** (ou la trajectoire) qui commence à $\underline{x}^{(0)}$. La durée entre deux réinitialisations successives sera appelée ***itération***.

L'idée de base de la modélisation est résumée ci-après.

Ce que nous intéresse particulièrement est la liaison entre les positions des ouvriers de la fin de chaque itération et celles du commencement de la suivante. Cette liaison sera décrite par une fonction notée par *f*.

Cette liaison impose la *rythmicité* du fonctionnement, ce qui s'exprime par *le temps de cycle* ou par *le taux de production*.

Comme suite, l'effort de modélisation sera concentré sur l'évolution de l'orbite en fonction de la variable t : $\left\{\underline{x}^{(t)}\right\}_{t=0,1,2,...}$. Nous ne désirons pas pour l'instant formaliser exactement l'évolution dans le cas général des positions des ouvriers *pendant* la durée d'une itération. Nous verrons plus loin qu'un tel modèle est aisément déductible dans un cas particulier.

Donc, nous nous intéressons à l'évolution du vecteur

$$\underline{x}^{(t)} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \\ \dots \\ x_n^{(t)} \end{bmatrix}_{t=0,1,2,...},$$

qui vérifie les relations :

$$\underline{x}^{(t+1)} = f(\underline{x}^{(t)}),\ t = 0,1,... \qquad \Leftrightarrow \qquad \underline{x}^{(t)} = f^t(\underline{x}^{(0)}),\ t = 0,1,...$$

La fonction f décrit la liaison entre tous deux vecteurs de positions successives de réinitialisation. L'existence d'un comportement stabilisé, périodique de durée égale au temps de cycle, signifie que, à partir d'une certaine itération, si les ouvriers commencent à certaines positions, ils vont toujours réinitialiser aux mêmes positions.

La conclusion qui s'impose est que :

> L'existence d'un ***comportement stabilisé*** pour une ligne TSS est équivalente à l'existence d'***un point fixe*** pour la fonction f.

Une fois que le modèle de la ligne a été déduit, il sera exploité pour l'analyse. Nous allons présenter les plus importants résultats théoriques, issus d'une approche stochastique basée sur le modèle ci-dessus, dont les démonstrations se trouvent en Annexe A.

V.2.3 Principaux résultats théoriques [BART 96a]

V.2.3.1 Convergence vers le point fixe

L'analyse des lignes TSS repose sur l'analyse des propriétés de la fonction f. Nous considérons que les plus importants points de l'analyse sont les conditions d'***existence*** et d'***unicité du point fixe*** de la fonction f – qui correspond à la ***possibilité*** d'un comportement stabilisé, désirable du point de vue de la conception – aussi bien que la ***convergence*** vers le point fixe, qui ***garantit*** le fonctionnement stabilisé, avec une périodicité optimale.

Le premier théorème établit l'existence d'un point fixe pour toute ligne TSS, quelle que soit sa configuration particulière.

THÉORÈME T-V.1 : *existence du point fixe*

> Pour toute ligne TSS **il existe *un point fixe***, c'est à dire il existe des positions des ouvriers, telles que, si les ouvriers commencent aux positions $\underline{x}^*$, alors ils réinitialisent toujours aux positions $\underline{x}^*$:
>
> $\underline{x}^* = f(\underline{x}^*)$

Le point fixe est un point d'équilibrage. Ce théorème affirme que *l'équilibrage* est toujours – c'est à dire, indifféremment de la configuration de la ligne – *au moins théoriquement possible*.

La démonstration de ce théorème est basée sur le raisonnement par l'absurde, à partir d'une expérience imaginaire, de deux lignes absolument pareilles, mais supposées se comporter différemment (voir Annexe A).

L'énoncé du premier lemme exprime ***une condition suffisante*** pour l'unicité du point fixe. Nous allons voir que cette condition est l'expression d'*une contrainte technologique*, qui permet, dans une démarche de conception, d'imposer d'une manière très simple, mais aussi bien nuancée, une certaine valeur du temps de cycle.

Lemme L-V.1 : *unicité du point fixe*

> Si les ouvriers sont ordonnés ***du plus lent au plus rapide*** sur la ligne, alors ***le point fixe est unique***.

Le lemme ci-dessus exprime la condition du "*bon rangement*" des ouvriers sur la ligne comme étant suffisante pour l'existence *potentielle* d'un comportement stabilisé, caractérisé par *une valeur unique du temps de cycle*, qui ne dépend pas d'autres conditions. Le bon rangement est compris dans l'esprit de la relation (V.1).

Remarquons que, si cette condition n'est pas remplie, on peut avoir – mais pas forcément – *plusieurs points fixes* (voir aussi les cas de figures de simulation, dans V.2.4.2). Ce qui signifie que, dépendant d'autres éléments, le comportement stabilisé sera caractérisé par des différentes valeurs du temps de cycle.

Le deuxième théorème montre que le rangement des ouvriers du plus lent au plus rapide garantit le fait que le comportement périodique est *effectivement atteint*.

THÉORÈME T-V.2 : *convergence vers le point fixe unique*

> Pour toute ligne TSS, si les ouvriers sont rangés du plus lent au plus rapide, alors toute orbite (trajectoire) des positions des ouvriers ***converge*** vers l'unique point fixe.

La démonstration de ce théorème a besoin de quelques **notations préliminaires**.

Notons par $\tau_i(x,x') = \int_x^{x'} \frac{dz}{v_i(z)}$ le temps nécessaire à l'ouvrier i pour se déplacer à partir de la position x jusqu'à la position x', $0 \le x \le x' \le 1$. Ce qui est une fonction :

- strictement croissante en x' : $x_1' < x_2' \Rightarrow \tau(x,x_1') < \tau(x,x_2')$
- strictement décroissante en x : $x_1 < x_2 \Rightarrow \tau(x_1,x') > \tau(x_2,x')$

Observation :

Compte tenu de cette notation, dans une interprétation systémique, les ouvriers peuvent être considérés comme "intégrateurs" des vitesses de travail. Cette observation nous aidera à construire un schéma de simulation d'une ligne TSS (voir V.2.4).

Notons par $P_k = \sum_{j=0}^{k} p_j$ le travail cumulé sur un article dès qu'il quitte le poste k, $k=1,2,...m$, $p_0=0$. Vue cette notation, l'intervalle (P_{k-1},P_k) représente le contenu de travail du poste k. Lorsqu'on impose qu'un seul ouvrier peut travailler sur un certain poste à un moment donné, il n'est pas possible que deux x_i-s se trouvent dans le même intervalle (P_{k-1},P_k).

Afin d'éclaircir la situation d'un ouvrier situé à la frontière d'un poste, nous définissons :

$$\underline{\underline{x}} = P_{k-1}\text{, si } x \in [P_{k-1},P_k) \text{ ;} \qquad \overline{\overline{x}} = P_k\text{, si } x \in [P_{k-1},P_k).$$

Nous pouvons regarder l'intervalle $[x_i^{(t)},x_{i+1}^{(t)}]$ comme *une partition dynamique* du contenu de travail, qui est affectée à l'ouvrier *i* pendant la *t*-ième itération. Cet intervalle représente ***la partie de travail attribuée*** à l'ouvrier, qui est *éventuellement* exécutée effectivement (en fait, cette partie est effectivement exécutée dès que la ligne atteint le régime stabilisé).

Nous appelons **allocation** – notée par $a_i^{(t)}$ – le temps nécessaire à l'ouvrier *i* pour qu'il finisse sa partie de travail *suggérée* pendant la *t*-ième itération, y compris le temps de travail effectif et les éventuels délais à cause du blocage à l'entrée sur des postes occupés (voir Annexe A pour les propriétés des allocations). Il existe *deux sortes* d'allocations :

- allocation *simple* – entièrement formée de temps de travail effectif – exprimée par :

  $$a_i^{(t)} = \tau_i(x_i^{(t)},x_{i+1}^{(t)}),\ \ i=1,2,...n\ ;$$

- allocation *retardée* – qui inclut également des délais – exprimée par :

  $$a_i^{(t)} = \tau_i(\underline{\underline{x_{i+1}^{(t)}}},x_{i+1}^{(t)}) + \tau_{i+1}(x_{i+1}^{(t)},\overline{\overline{x_{i+1}^{(t)}}}),\ \ i=1,2,...n\,.$$

Remarquons que l'allocation du dernier ouvrier, lorsqu'il n'est jamais bloqué, est toujours une allocation simple. Le comportement général de la ligne est décrit par les équations ci-dessous :

$$\begin{cases} a_n^{(t)} = \tau_n(x_n^{(t)},1) \\ a_i^{(t)} = \tau_i(x_i^{(t)},x_{i+1}^{(t)}) + max\left\{0, \tau_{i+1}(x_{i+1}^{(t)},\overline{\overline{x_{i+1}^{(t)}}}) - \tau_i(x_i^{(t)},\underline{\underline{x_{i+1}^{(t)}}})\right\}, & i=1,2,...n-1 \end{cases}$$

Ce théorème a une démonstration élaborée (voir Annexe A pour les grandes lignes).

Le théorème suivant est une forme particulière du deuxième théorème. L'hypothèse supplémentaire adoptée est la modélisation des ouvriers par **fonctions constantes de vitesse** : $v_i(x) \equiv v_i = constante,\ i=1,2,...n$. Cette hypothèse simplificatrice est valable surtout pour les lignes qui requirent *presque la même habileté de travail pour toutes les opérations*. Ce qui est vrai pour la plupart des lignes TSS. Rappelons la notion de "**taux de production**", exprimé :

- soit par *le nombre d'articles* par unité de temps,
- soit par le temps consommé pour finir un article (*le temps de cycle*).

THÉORÈME T-V.3 :

Si les vitesses sont *constantes* $v_i(x)=v_i$ et $v_1<v_2<....<v_n$ et si, en plus, les ouvriers ne sont *jamais bloqués*, alors la ligne converge *exponentiellement* vers *l'unique point fixe* pour lequel :

- l'ouvrier *i* répète l'exécution d'une portion de travail $\left[\dfrac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}, \dfrac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j}\right]$;

- le taux de production est $\sum_{j=1}^{n} v_j$, le meilleur possible.

Nous remarquons qu'une autre supposition supplémentaire est l'inexistence des blocages, ce qui signifie que toutes les allocations sont *simples*. Dans les conditions du théorème T-V.3, le comportement de la ligne est :

$$(V.2) \qquad \begin{cases} a_1^{(t+1)} = a_n^{(t)} \\ a_i^{(t+1)} = \dfrac{v_{i-1}}{v_i} \cdot a_{i-1}^{(t)} + (1 - \dfrac{v_{i-1}}{v_i}) \cdot a_n^{(t)}, \quad i=1,2,...n, \end{cases}$$

équations qui représentent **un système dynamique linéaire** :

$$(V.3) \qquad \underline{a}^{(t+1)} = \mathbf{T} \cdot \underline{a}^{(t)},$$

où $\mathbf{T}$ est *la matrice de transition d'une chaîne de Markov* finie, irréductible et apériodique. Nous avons noté par $\underline{a}^{(t)}$ le vecteur des allocations à l'itération t.

La relation (V.3) assure la convergence de la suite $\{a_n^{(t)}\}_{t=0}^{\infty}$ et, ensuite, la convergence de l'orbite $\{\underline{x}^{(t)}\}_{t=0}^{\infty}$.

Le temps de cycle est donné par la plus large des allocations des ouvriers, c'est à dire par *l'allocation du dernier ouvrier* (l'ouvrier n). Ce résultat montre que "le bon rangement" des ouvriers assure que la plus large allocation converge et, de plus, elle diminue après chaque suite de n itérations. Comme suite, le taux de production augmente vers une limite qui ne dépend pas des positions initiales des ouvriers sur la ligne.

L'atteinte d'un temps de cycle constant et, de plus, minimum, caractérise le **comportement stabilisé**. Remarquons que la condition exprimée par le théorème ci-dessus est seulement suffisante pour garantir le comportement stabilisé. Il existe des cas où ce comportement est atteint même si les ouvriers ne sont pas rangés du plus lent au plus rapide (voir l'analyse par simulation – V.2.4.2).

La première démonstration de ce théorème, telle qu'elle est donnée dans [BART 96a], utilise l'approche stochastique (voir Annexe A). Au paragraphe suivant de ce chapitre, nous allons présenter une nouvelle démonstration de ce théorème, cette fois-ci dans un contexte déterministe, en modélisant une ligne TSS comme système dynamique linéaire discret.

V.2.3.2 Le comportement "compliqué" et le comportement "anomal"

Intuitivement, il serait évident que l'accroissement du nombre des ouvriers ou l'augmentation des vitesses de certains ouvriers auraient comme conséquence l'augmentation du taux de production. Mais il existe des cas où cette affirmation n'est pas valable et où, en outre, le taux de production diminue. Ces cas s'appellent ***cas anomaux***.

Nous revenons au cas des vitesses de travail quelconques. Le théorème suivant garantit le comportement sans "anomalies" dans le cas où les ouvriers restent ordonnés en bon ordre.

THÉORÈME T-V.4 :

> Si $v_1 \prec v_2 \prec ... \prec v_n$, alors, en ajoutant ou en accélérant un ouvrier, il ne se produit jamais de diminution du taux de production.

Observation :

Malheureusement, il n'existe aucune démonstration pour la supériorité d'une ligne avec la bonne configuration, dans le cas où les vitesses ne sont pas forcément constantes. La raison est que la logique de TSS ne peut pas garantir par elle-même le meilleur taux de production

s'il existe *une discordance entre l'ordonnancement des ouvriers et l'affectation du contenu de travail aux postes*. Néanmoins, une ligne TSS ainsi structurée aura *toujours* un taux de production qui est bon au sens suivant : une autre séquence d'ouvriers peut se comporter plus mal, mais pas trop mieux.

Le théorème suivant garantit un bon comportement – au sens du taux de production – d'une ligne TSS où le dernier ouvrier est le plus rapide.

THÉORÈME T-V.5 :

> Quand *le dernier ouvrier est le plus rapide*, le taux de production de la ligne à son point fixe est toujours au plus *n* fois plus grand que le meilleur obtenu par une autre séquence d'ouvriers.

Le comportement compliqué d'une ligne TSS englobe les cas où la condition $v_1 \prec v_2 \prec \ldots \prec v_n$ ***n'est pas remplie***. Les résultats de simulation (voir V.2.4.2) nous suggèrent que ce type de comportement est très proche du comportement dans l'espace d'états des *systèmes dynamiques non linéaires*.

Rappelons que nous sommes intéressés par l'orbite – ou *la trajectoire* – de la ligne, c'est à dire par l'évolution temporelle des positions successives de réinitialisation. Cette trajectoire possède, dans ce cas, des propriétés semblables à celles des trajectoires d'état des systèmes dynamiques non linéaires.

Nous énumérons ci-dessous quelques exemples de comportements compliqués :

- l'existence d'un point fixe "*instable*" – attractif ou répulsif – quand les orbites s'obtiennent comme *des cycles limites* ;
  exemple : lorsqu'un ouvrier plus rapide est systématiquement bloqué par un autre, plus lent, la ligne atteint un comportement stabilisé, mais sous-optimale du point de vue de l'équilibrage (elle ne fonctionne pas au temps de cycle minimum) ;
- le comportement *quasi-périodique* (prédictible, mais pas périodique), où les positions de réinitialisation ne sont jamais les mêmes d'une itération à la suivante ;
  exemple : lorsque le premier et le dernier ouvrier ont la même vitesse, $v_1=v_n$, et le contenu total de travail est distribué en mode égal aux postes, les orbites convergent vers la périphérie d'une ellipse ;
- la présence de *points fixes multiples*, ou de *plusieurs cycles limites* ;
- le comportement à la longue qui *dépend des positions initiales* des ouvriers sur la ligne.

V.2.3.3 Quelques points faibles du modèle

Le principal point faible du modèle présenté ci-dessus est **la modélisation** des ouvriers **en fonction de la vitesse de travail**. Comme nous l'avons montré, cela est une hypothèse simplificatrice, adoptée pour une compréhension de principe du fonctionnement d'une ligne TSS, en ignorant les détails qui concernent, par exemple : l'affectation des tâches aux postes de travail, la chorégraphie détaillée des mouvements des ouvriers, leurs motivations, etc.

Les auteurs cités considèrent qu'un autre modèle, plus nuancé, des ouvriers pourrait être une *fonction de vitesse concave et unimodale*.

Dans l'ensemble des hypothèses qui ont permis le développement de ce modèle on considère implicitement que les **temps opératoires** sont **déterministes**. L'idée d'introduire un élément aléatoire dans le temps de travail consommé sur un poste a été considérée ultérieurement, par les mêmes auteurs, sous la forme équivalente de considérer la variabilité

du contenu de travail. Dans [BART 99a] on montre qu'une ligne TSS est efficace même si on considère **un modèle stochastique du travail**.

Il n'existe pas une formule exacte de détermination du taux de production pour le cas général, lorsque nous ne savons pas précisément quels éléments l'influencent.

De même, le comportement "compliqué" présente un intérêt particulier. Son interprétation est rendue plus facile par la simulation, mais nous devons attentivement interpréter les résultats de simulation dans ce cas. Ce comportement n'a pas été entièrement compris, même dans les cas les plus simples, par exemple, pour les lignes avec deux ou trois ouvriers [BART 99b].

Dans ce qui suit nous allons présenter *le principe d'une analyse par simulation* des lignes TSS, qui repose sur la ressemblance de ces lignes avec les *systèmes dynamiques non linéaires*.

V.2.4 Analyse par simulation

Dans une première approche, nous pouvons modéliser une ligne TSS en tant que système dynamique non linéaire, afin de construire ensuite **un schéma non linéaire de simulation**. Nous voulons mettre en évidence non seulement l'évolution des positions de réinitialisation, mais également le mouvement des ouvriers ***pendant*** la durée d'une itération. Dans l'annexe B nous présentons une implantation de ce schéma en SIMULINK.

Nous avons fait une campagne de test pour quelques cas de figures importants, sur une configuration de *trois ouvriers*, chacun étant modélisé par une *fonction de vitesse constante*. C'est la configuration qui nous permet l'interprétation la plus intuitive des résultats de simulation.

V.2.4.1 Construction du schéma de simulation

Tout d'abord, nous analysons les deux étapes génériques du fonctionnement d'une ligne TSS : *le moment de réinitialisation* et *l'itération* (l'intervalle temporel d'une réinitialisation à la suivante).

- Pendant une itération, chaque ouvrier i, $i=1,2,...n-1$, avance normalement à sa vitesse de travail, v_i. L'exception survient à la frontière d'un poste qui est occupé par l'ouvrier suivant. En ce cas-là, l'ouvrier doit s'arrêter, ce qui signifie que sa vitesse devient nulle (excepté le dernier ouvrier). Donc, la position d'un ouvrier sur la ligne s'obtient en principe comme *l'intégrale* de sa vitesse.
  La conclusion est que pendant une itération les ouvriers fonctionnent comme "**intégrateurs**" de leurs vitesses de travail.

- La réinitialisation signifie que chaque ouvrier change ***instantanément*** sa position sur la ligne et puis il recommence à travailler à sa vitesse de travail. La réinitialisation se produit ***dans le même instant*** pour tous les ouvriers : quand le dernier ouvrier finit son travail. Donc, elle surviendra dans le schéma de simulation lors de la décision d'un *comparateur*. Cette décision doit réinitialiser les intégrateurs qui représentent les ouvriers.

Nous voulons **un modèle unique** du comportement des ouvriers. Une conclusion se dégage lors de cette courte analyse : les ouvriers peuvent être modélisés comme **intégrateurs réinitialisables** de leurs vitesses de travail.

Revenons maintenant aux moments d'arrêt des ouvriers, qui peuvent survenir au cours d'une itération. L'arrêt d'un ouvrier *i*, *i=1,2,…n-1*, est équivalent au changement brusque de sa vitesse de v_i à *0*, quand l'ouvrier *a l'intention* de continuer le travail sur un poste, mais ce poste-là est occupé par l'ouvrier *i+1*. Le redémarrage d'un ouvrier correspond à la situation inverse. Ces deux phénomènes seront modélisés par un *relais*.

Il est nécessaire que le schéma de simulation contienne *un bloc non linéaire* qui détermine quel est le poste occupé à un moment donné par un certain ouvrier, en utilisant la valeur de sa position instantanée. Ce bloc sera noté par **B**. Il est décrit par la fonction $f_{\mathbf{B}}$.

Le schéma de modélisation d'un ouvrier est présenté dans la figure V-2.

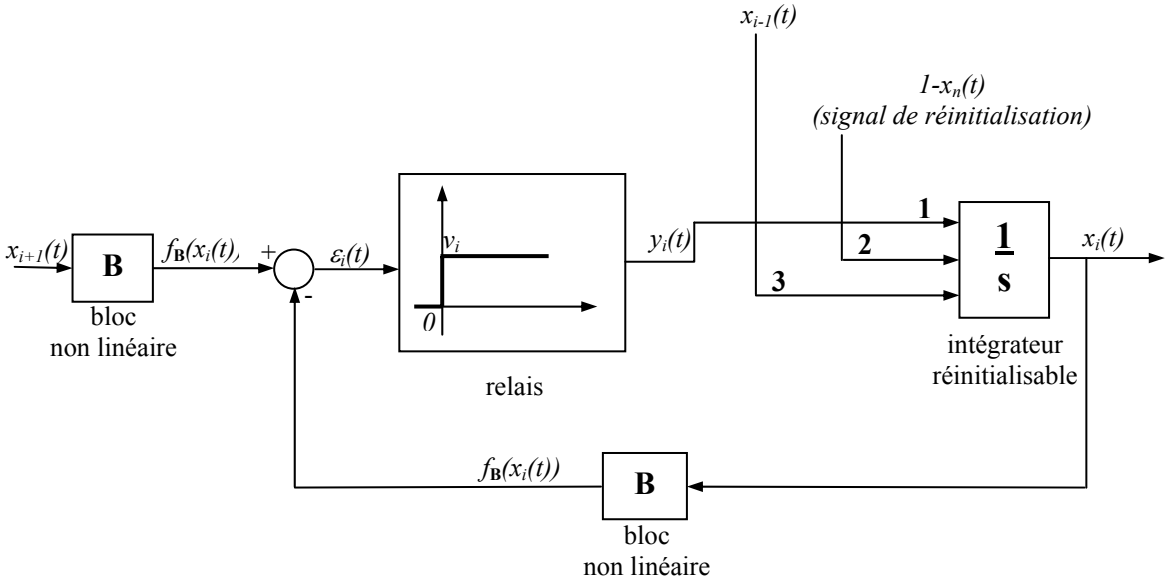


Fig. V-2. Le schéma de modélisation de l'ouvrier *i*, *i=1,2,…n-1*

Soit *i* de {*1,2,…n-1*} fixé.

Ensuite, nous allons détailler le fonctionnement de tous les blocs qui interviennent dans le schéma de modélisation de l'ouvrier *i*.

Le fonctionnement du relais

L'expression analytique du fonctionnement du relais est :

$$(V.4) \qquad y_i(t) = \frac{v_i}{2} + \frac{v_i}{2} \cdot sign(\varepsilon_i(t)),$$

où la fonction *sign*, comme il est bien connu de l'automatique non linéaire, modélise le relais symétrique :

$$sign(x(t)) = \begin{cases} 1, & x(t) > 0 \\ -1, & x(t) \le 0 \end{cases}$$

Le fonctionnement de l'intégrateur réinitialisable

Ce bloc a trois entrées, conformément à la figure V-2. Sa sortie est l'intégrale du signal **1**, tant que le signal **2** est positif. C'est à dire, tant que $x_n(t)<1$ – autrement dit, tant que le dernier ouvrier n'a pas encore fini son travail – la position d'un ouvrier quelconque s'obtient comme l'intégrale de sa vitesse, qu'elle soit positive ou nulle.

Quand le signal **2** devient nul, l'état initial de l'intégrateur est réinitialisé à la valeur du signal **3**. C'est le moment où chaque ouvrier bouge en arrière jusqu'à la position de son prédécesseur. Conformément aux règles TSS, le signal **2** redevient immédiatement positif et, donc, le bloc recommence à intégrer le signal **1**.

Comme conclusion, en gardant les notations de la figure V-2, le fonctionnement de l'intégrateur réinitialisable est décrit analytiquement par la suite :

$$\text{(V.5)} \qquad \begin{cases} Si \quad x_n(t) < 1, \quad \dot{x}_i(t) = y_i(t) \\ sinon, \qquad x_i(t^{+0}) = x_{i-1}(t) \end{cases},$$

où la notation "t^{+0}" signifie le moment immédiatement après le moment "t".

Le fonctionnement du bloc **B**

Afin de détailler l'expression mathématique de la fonction $f_{\mathbf{B}}$, rappelons tout d'abord la notation $P_k = \sum_{j=0}^{k} p_j$, qui signifie le travail cumulé sur un article dès qu'il quitte le poste k, $k=1,2,...m$, $p_0=0$. L'entrée du bloc **B** est $x_i(t)$, la position instantanée de l'ouvrier i. La sortie de ce bloc doit être une des valeurs P_l, $l=1,2,...m$. Ce qui signifie que l'ouvrier i travaille sur le poste l au moment t. Il en résulte que la fonction $f_{\mathbf{B}}$ peut être représentée graphiquement comme illustré dans la figure V-3.

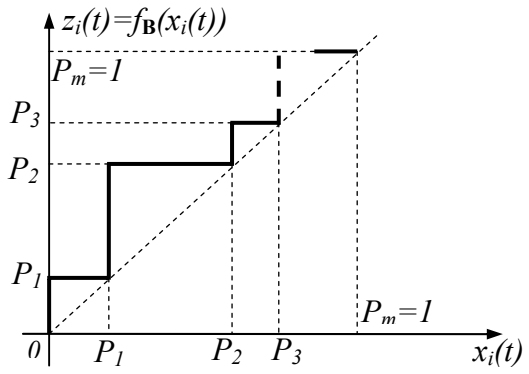


Fig. V-3. Représentation graphique de la fonction $f_{\mathbf{B}}$ du bloc non linéaire **B**

Un schéma qui réalise la fonction $f_{\mathbf{B}}$ peut être conçu comme *une connexion parallèle de m relais*, comme illustré dans la figure V-4. En gardant les notations de cette figure, il en résulte que l'expression analytique de la fonction $f_{\mathbf{B}}$ est :

$$\text{(V.6)} \qquad f_{\mathbf{B}}(x_i(t)) = \frac{1}{2} + \frac{1}{2} \cdot \sum_{k=1}^{m} \{p_k \cdot sign(x_i(t) - P_{k-1})\}$$

Nous pouvons maintenant donner **le modèle analytique non linéaire d'une ligne TSS**, qui a été implémenté en Simulink par un diagramme de blocs non linéaires permettant l'analyse par simulation de ce type de lignes (voir Annexe B).

Dans ce but, revenons à la modélisation d'un ouvrier i fixé (voir figure V-2). L'expression du signal $\varepsilon_i(t)$ est :

$$\varepsilon_i(t) = f_{\mathbf{B}}(x_{i+1}(t)) - f_{\mathbf{B}}(x_i(t))$$

Nous utilisons la relation (V.5), où le signal $y_i(t)$ s'exprime en utilisant la relation (V.4), et l'expression du signal $\varepsilon_i(t)$:

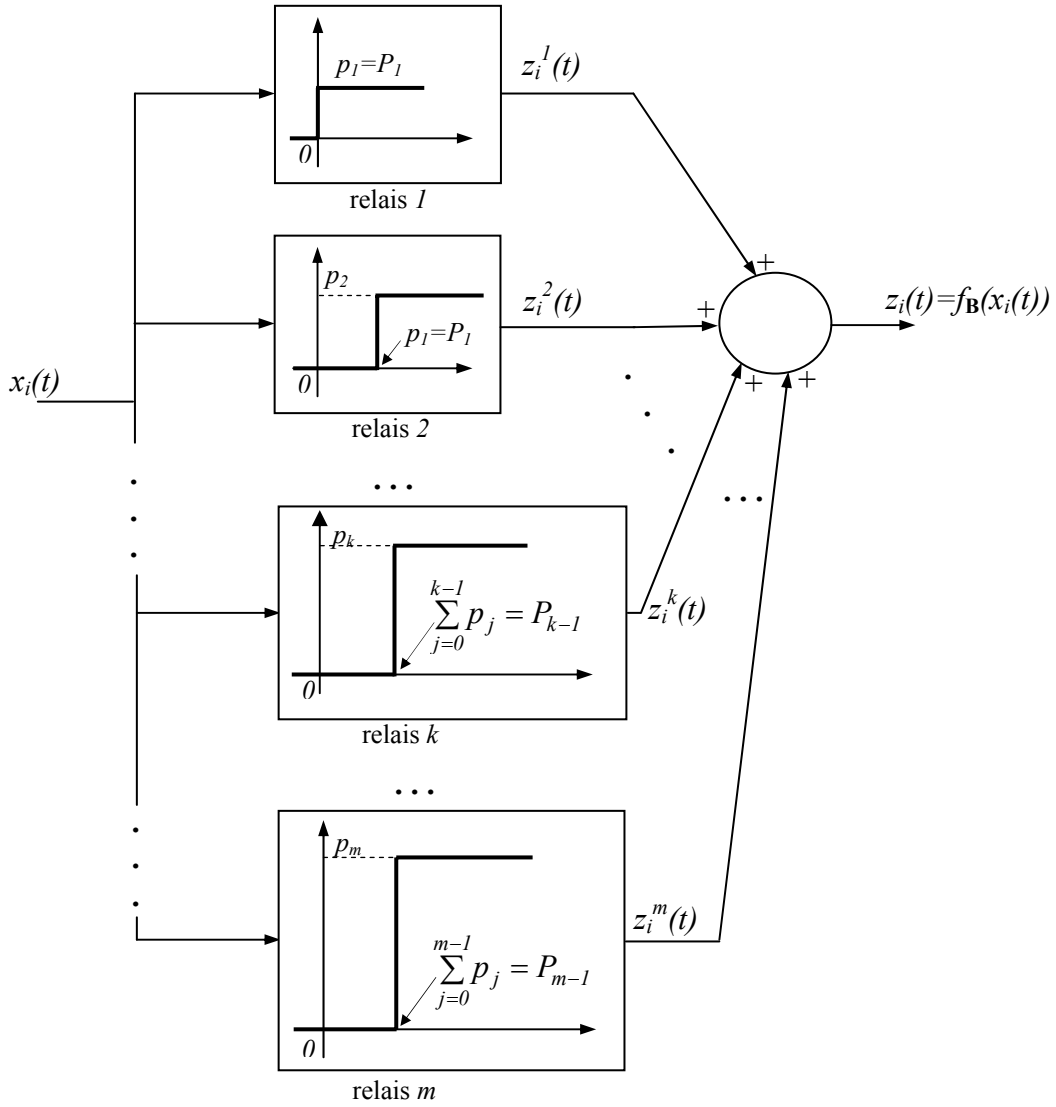


Fig. V-4. Le schéma du bloc non linéaire **B**

$$\begin{cases} Si \quad x_n(t) < 1, \quad \dot{x}_i(t) = \dfrac{v_i}{2} + \dfrac{v_i}{2} \cdot sign\{f_{\mathbf{B}}(x_{i+1}(t)) - f_{\mathbf{B}}(x_i(t))\} \\ \quad sin\,on, \qquad x_i(t^{+0}) = x_{i-1}(t) \end{cases}$$

En utilisant ensuite la relation (V.6), il s'ensuit que le comportement d'une ligne TSS formée de *n* ouvriers est décrit par le suivant **système d'équations différentielles** :

$$(V.7)\ \begin{cases} Si\ x_n(t) < 1, \begin{cases} \dot{x}_i(t) = \dfrac{v_i}{2} + \dfrac{v_i}{2} \cdot sign\left\{\dfrac{1}{2} \cdot \sum_{k=1}^{m} \{p_k \cdot (sign(x_{i+1}(t) - P_{k-1}) - sign(x_i(t) - P_{k-1}))\}\right\} \\ \qquad i = \overline{1, n-1} \\ \dot{x}_n(t) = v_n \end{cases} \\ \quad sin\,on, \quad \begin{cases} x_1(t^{+0}) = 0 \\ x_i(t^{+0}) = x_{i-1}(t), \qquad i = \overline{2, n} \end{cases} \end{cases}$$

V.2.4.2 Les principaux cas de figures

Le comportement d'une ligne TSS dépend de *deux paramètres* qui caractérisent la configuration de la ligne : les vitesses des ouvriers, $\{v_i(x)\}_{i=1,2,...n,\ x\in[0\,;1]}$, et l'affectation du contenu de travail aux postes, $\{p_k\}_{k=1,2,...m}$.

Dans tous les cas, nous avons considéré que les vitesses de travail sont **constantes** à travers la ligne. En fait, l'influence des vitesses se manifeste plutôt au niveau de leur *ordre relatif*, et pas forcément au niveau de leurs valeurs absolues. Ce qui nous est suggéré par les résultats théoriques listés auparavant.

Comme suite, la simulation est faite sur les deux axes de variation des paramètres susmentionnés, comme montré à la figure V-5.

vitesses / ***postes de travail***		DIFFÉRENTES			ÉGALES
		ordre quelconque		"bon ordre" respecté	
		$v_1 \neq v_n$	$v_1 = v_n$	$(v_1 < v_2 < \ldots < v_n)$	
STRICTEMENT DÉLIMITÉS	inégaux	***a)***	***b)***	***c)***	***d)***
	égaux	***e)***	***f)***	***g)***	***h)***
SANS POSTES		***i)***	***j)***	***k)***	***l)***

Fig.V-5. Synthèse des principaux cas de figures de simulation

Remarques :

1) Les cas "sans postes" correspondent en fait à une modification de la règle TSS. Plus précisément, il s'agit du relâchement de la règle de priorité sur un poste occupé, qui implique *l'inexistence des blocages*. Dans ce cas, la règle TSS ne peut pas gouverner les situations où le bon ordre n'est pas respecté, car le comportement d'un ouvrier plus rapide qui rattrape un autre, plus lent, n'est pas spécifié. C'est pourquoi nous avons renoncé aux cas ***i)*** et ***j)***, qui sont infaisables. Nous remarquons que la priorité est cependant implicitement respectée dans les cas ***l)*** et ***k)***.

2) Pour les autres cas où le bon ordre n'est pas respecté – ***a)***, ***b)*** ***e)*** et ***f)*** – la convergence des positions de réinitialisation vers le point fixe n'est pas garantie (les conditions du théorème T-V.2 ne sont pas remplies). Mais la ligne *peut atteindre* un régime stabilisé, dépendant du chargement des postes, $\{p_k\}_{k=1,2,...m}$. En général, il existe des blocages dans ces cas.

3) Les cas où les vitesses sont égales correspondent aux implantations avec des "*ouvriers standards*". Ils sont désirables en pratique.

Le schéma de simulation implémenté en Simulink est donné dans l'annexe B. Nous avons choisi une configuration de *trois ouvriers*, puisqu'elle permet la représentation de l'orbite comme une courbe plane décrite par $x_3(k)=g(x_2(k))$ (la règle TSS nous dit que $x_1(k)=0$, sauf éventuellement à la première itération), que nous appelons *la trajectoire d'état*.

Nous présentons ensuite quelques résultats de simulation obtenus sur ce schéma. Ils sont adressés en correspondance avec le tableau de la figure V-5. Nous utilisons les **notations** suivantes :

$x_i(k)$ – la position de réinitialisation de l'ouvrier i à l'itération k (variable d'état)
$x_i(t)$ – la position instantanée de l'ouvrier i
$\underline{x_0}$ – le vecteur des positions initiales
$\underline{v}$ – le vecteur des vitesses
$\underline{p}$ – la notation vectorielle pour $\{p_k\}_{k=1,2,...m}$

c) Ce cas est illustré aux figures V-6 et V-7, qui montrent qu'une équipe d'ouvriers bien rangés atteint le régime stabilisé après quatre itérations. Nous avons choisi :

$\underline{x_0}$ = [*0.2 0.4 0.6*],
$\underline{v}$ = [*1 1.2 1.7*],
$\underline{p}$ =[*0.3 0.2 0.3 0.2*].

Conformément au théorème T-V.2, la ligne converge vers *le point fixe unique*, qui caractérise le régime stabilisé dans l'espace d'états. Ce point est [*0 0.186 0.5232*], comme montré à la figure V-7. Il vient que le temps de cycle est T_c=*0.2804*, le meilleur possible pour la configuration des postes donnée. Remarquons que le chargement choisi des postes détermine, en vertu de la règle TSS, que *le premier ouvrier soit bloqué* pendant chaque itération.

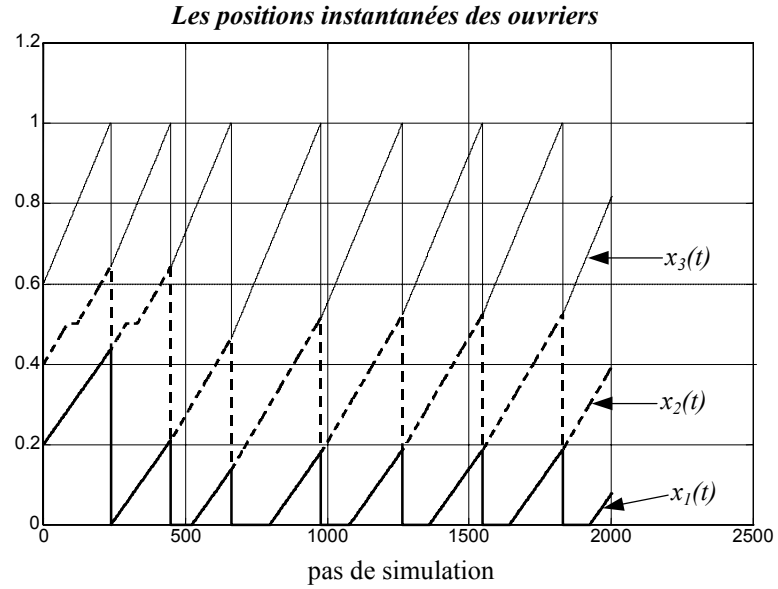


Fig.V-6. Exemple de comportement périodique *optimal* d'une ligne auto-équilibrée

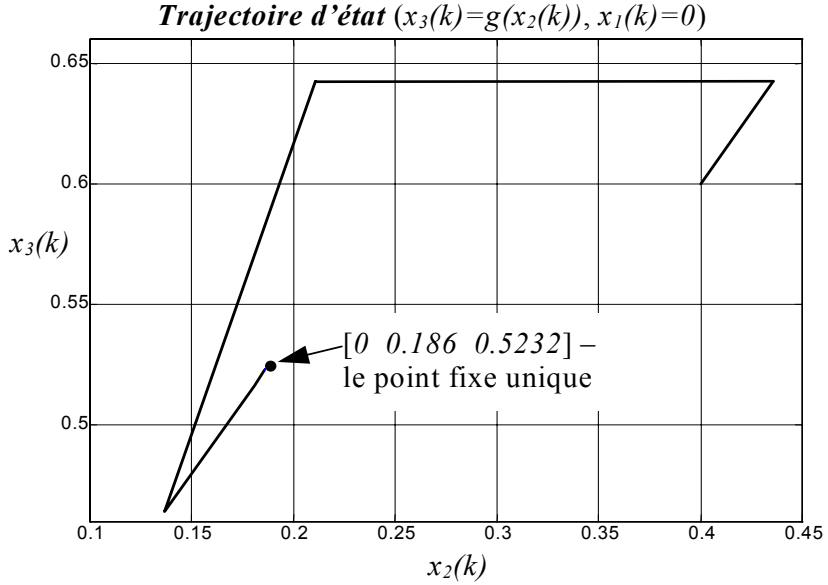


Fig.V-7. Convergence des positions de réinitialisation d'une ligne auto-équilibrée vers le point fixe unique

a) La convergence vers le point fixe n'est pas garantie. En général, une telle ligne se comporte périodiquement, mais avec une performance inférieure à celle mettant en œuvre le bon rangement. Nous pouvons dire que son équilibrage est *sous-optimal*.

Nous utilisons ce cas pour montrer qu'une même équipe d'ouvriers, rangés selon un ordre quelconque, peut atteindre des *points différents d'équilibrage*. Le point d'équilibre d'une telle ligne – et, donc, le temps de cycle – ne dépend pas des positions initiales des ouvriers, mais de l'affectation du travail aux postes, $\underline{p}$.

Nous avons choisi :

$\underline{x_0}$ = [*0.2 0.4 0.6*],

$\underline{v}$ = [*2.5 1 1.5*].

La figure V-8 présente l'évolution des positions instantanées pour $\underline{p}$=[*0.3 0.2 0.3 0.2*]. Le point d'équilibre est [*0 0.5025 0.6365*] et le temps de cycle T_c=*0.2423*. Ces valeurs ne changent pas lorsqu'on modifie les positions initiales.

Mais la situation change lorsqu'on change $\underline{p}$. Ce qui est montré à la figure V-9, pour $\underline{p}$=[*0.1 0.7 0.1 0.1*]. Les premiers ouvriers sont bloqués à l'entrée du deuxième poste. Par conséquent, le nouveau $\underline{p}$ a détérioré encore plus le comportement de la ligne : l'équilibrage est obtenu à [*0 0.1 0.233*] et T_c=*0.5113*.

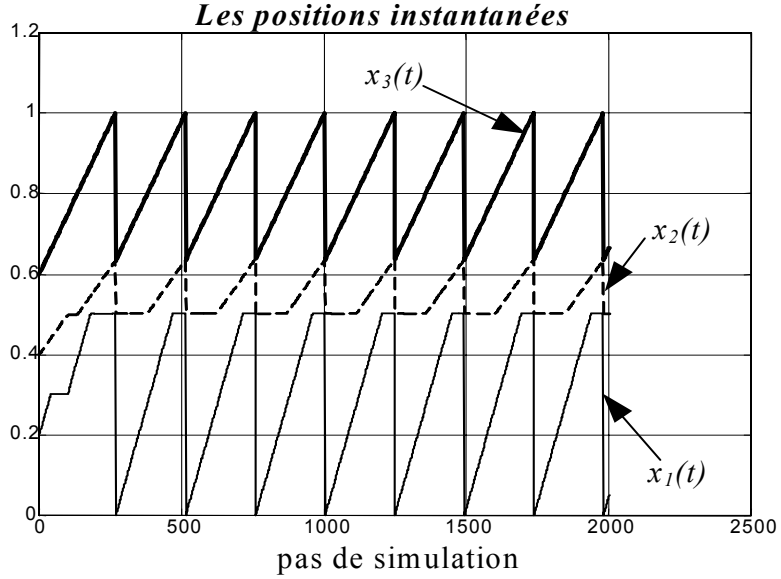


Fig.V-8. Exemple de comportement périodique *sous-optimal* d'une ligne auto-équilibrée

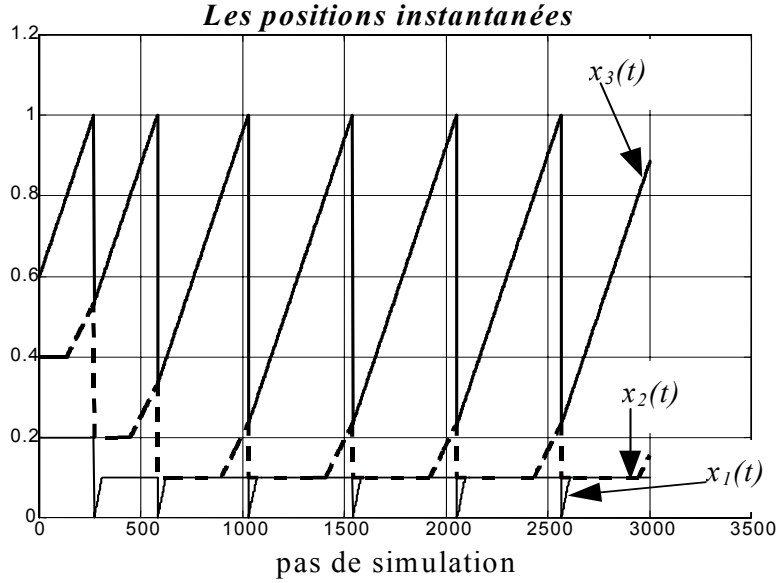


Fig.V-9. Un autre cas de comportement sous-optimal, pour la même ligne

k) Ce cas remplit *exactement* les conditions du théorème T-V.3 : le bon ordre et l'absence des blocages, assurée par l'absence des postes. Ce qui signifie que la ligne se comporte périodiquement, avec un *temps de cycle optimal*, c'est à dire le meilleur par rapport à tout choix de la configuration des postes, $\underline{p}$. Le point d'équilibre est le point fixe unique de la ligne, dont l'expression générale est :

$$\left[0 \quad \frac{v_1}{v_1 + v_2 + \ldots v_n} \quad \frac{v_1 + v_2}{v_1 + v_2 + \ldots + v_n} \quad \ldots \quad \frac{v_1 + v_2 + \ldots + v_{n-1}}{v_1 + v_2 + \ldots + v_n} \right].$$

Nous présentons à la figure V-10 les résultats de simulation d'une ligne avec :
$\underline{x_0}$=[*0.3 0.4 0.7*],
$\underline{v}$=[*3 4 8*],
dont l'équilibrage s'obtient à [*0 0.2 0.466*], ce qui correspond à l'expression ci-dessus. Le temps de cycle est T_c=*0.066*.

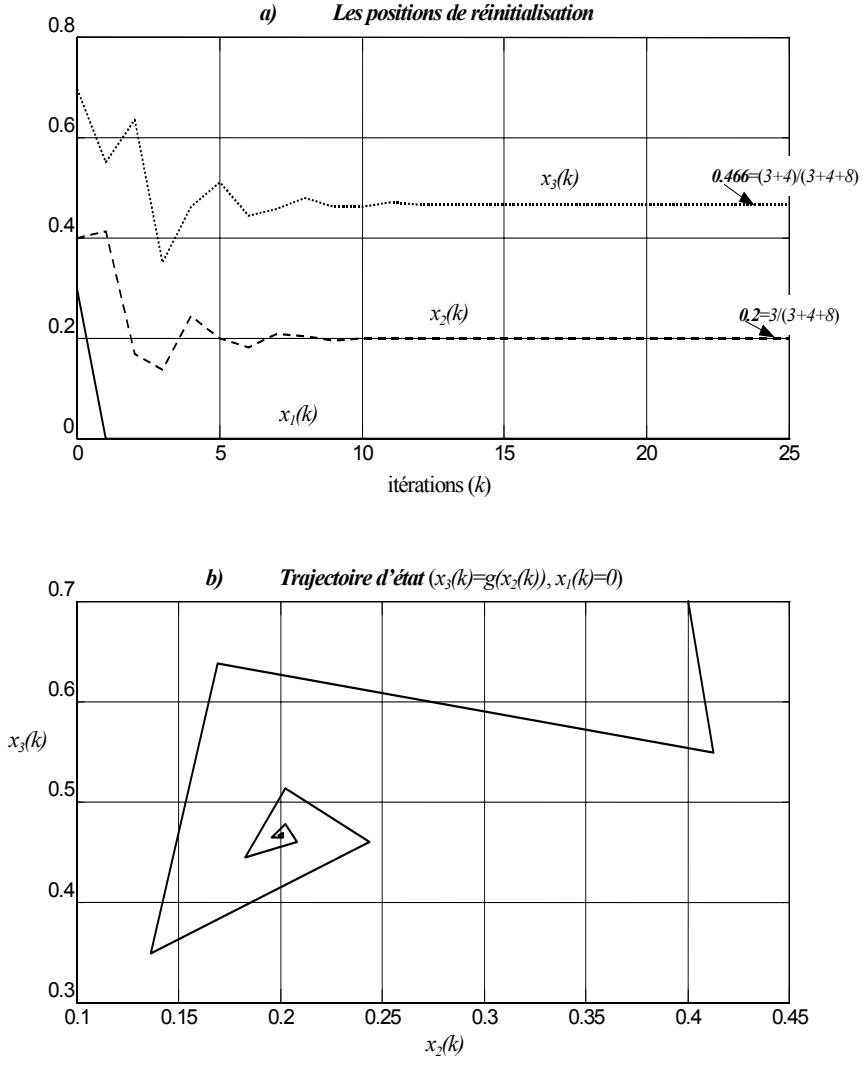


Fig.V-10. Convergence vers le comportement *optimal* d'une ligne auto-équilibrée *sans blocages*

l) Ce cas met en évidence une évolution de type *cycle limite* des positions de réinitialisation. L'égalité des vitesses et l'absence des postes implique l'absence des blocages.

On obtient un comportement qui n'est pas convergent vers un point fixe, mais vers une séquence de points. Donc, une telle ligne ne fonctionne pas à un taux de production constant.

Nous avons choisi :

$\underline{x_0}$=[*0.3 0.4 0.7*],

$\underline{v}$=[*3 3 3*].

Avec ces données, nous montrons dans la figure V-11 l'évolution des positions de réinitialisation des ouvriers et leur *cycle limite* mis en évidence sur la trajectoire d'état.

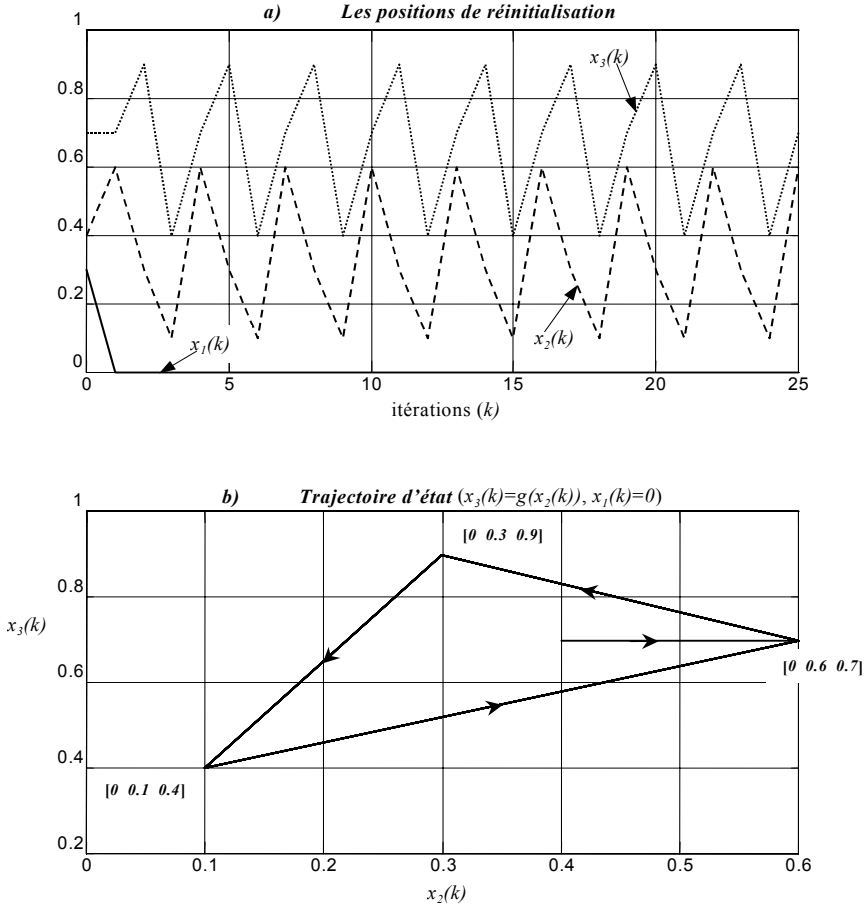


Fig.V-11. Comportement de type *cycle limite* d'une ligne auto-équilibrée sans blocages, avec "*ouvriers standards*"

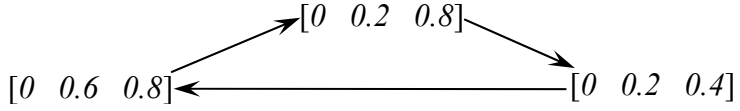
h) Par rapport au cas antérieur, nous introduisons un chargement égal des postes de travail. Ce qui ne change pas le comportement de la ligne, car l'absence des blocages se maintient. Pour une telle ligne, avec :

$\underline{x_0}$=[*0.2 0.4 0.6*],

$\underline{v}$=[*2 2 2*],

$\underline{p}$=[*0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1*],

nous présentons à la figure V-12 l'évolution des positions instantanées et les positions de réinitialisation, qui convergent sur le *cycle limite* :



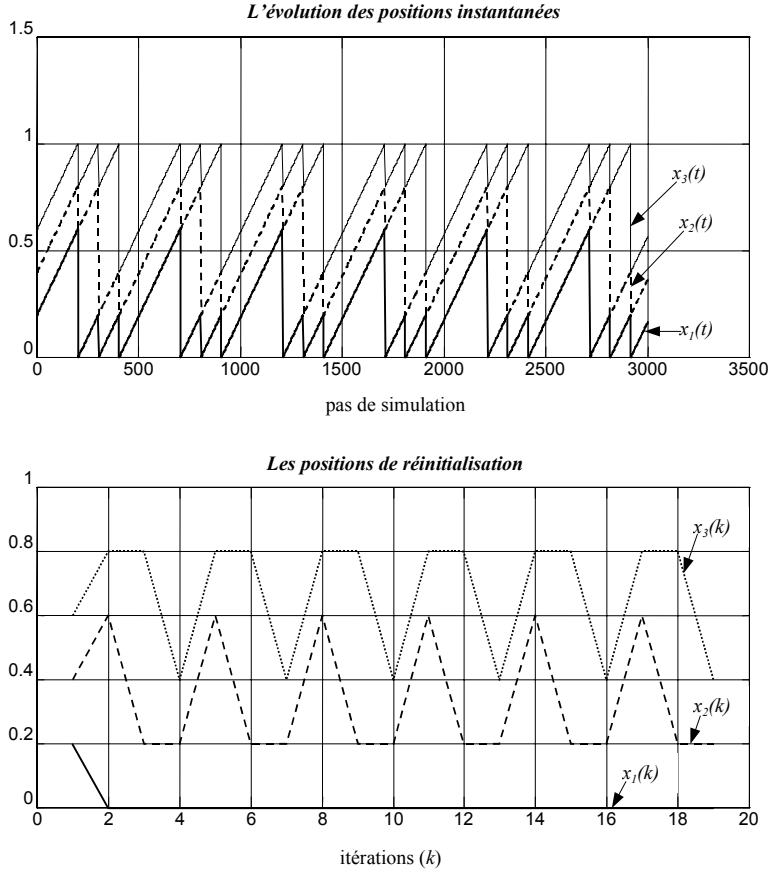


Fig.V-12. Un autre exemple de comportement de type *cycle limite* d'une ligne auto-équilibrée avec *ouvriers standards*

d) Si nous considérons des chargements différents des postes sur une ligne avec ouvriers standards (qui ont la même vitesse de travail), alors *la présence éventuelle des blocages impose un comportement périodique*, avec un taux de production constant. C'est le cas, par exemple, d'une ligne avec :

$\underline{x_0}$=[*0.05 0.65 0.9*],
$\underline{v}$=[*2 2 2*],
$\underline{p}$=[*0.1 0.6 0.1 0.2*],

où les premiers ouvriers sont bloqués pendant chaque itération, mais la ligne se comporte périodiquement. La figure V-13 montre que son point d'équilibre est [*0 0.1 0.4*] et, par conséquent, T_c=*0.3*. Comme nous l'avons montré aussi dans le cas ***a)***, un autre choix de $\underline{p}$ conduirait à une autre valeur du temps de cycle.

f) Si le premier et le dernier ouvrier travaillent à la même vitesse et les postes ont tous le même chargement, *toutes les orbites convergent à la périphérie d'une ellipse*. Les positions de réinitialisation ne se répètent jamais, ce que nous appelons *comportement quasi-périodique*, c'est à dire, prédictible, mais non périodique.

Nous illustrons ce cas par l'exemple suivant :

$\underline{x_0}$=[*0 0.4 0.6*],
$\underline{v}$=[*2.5 1 2.5*],
$\underline{p}$=[p_i], p_i=*0.05*, *i=1,2,…20*.

La figure V-14 présente la trajectoire d'état dans ce cas. Évidemment, un tel cas n'est pas désirable en pratique, puisque la variation du temps de cycle n'est même pas périodique.

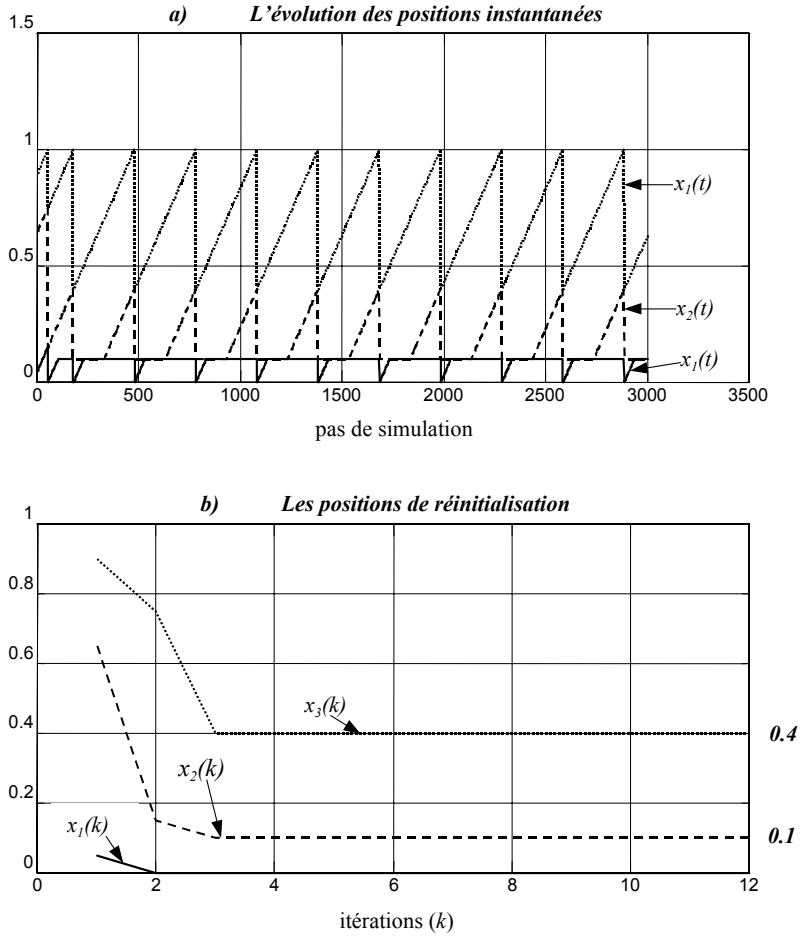


Fig.V-13. La présence des *blocages* sur une ligne auto-équilibrée avec *ouvriers standards* impose un *comportement périodique*

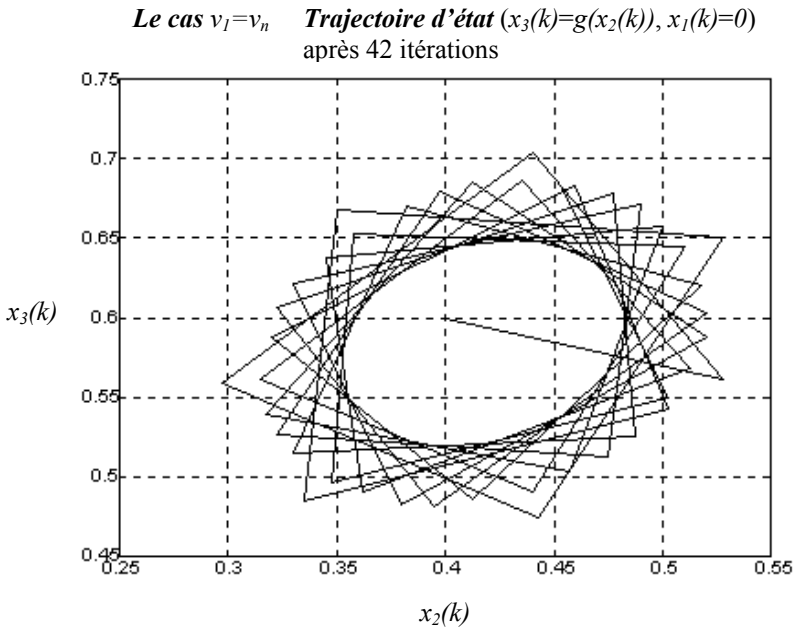


Fig.V-14. Exemple de comportement quasi-périodique d'une ligne auto-équilibrée ($v_1=v_n$)

V.2.5 Avantages des lignes auto-équilibrées du point de vue de la conception

La conception des lignes auto-équilibrées repose sur la valorisation de leur spécificité.

Grâce à la propriété d'auto-équilibrage, dans ce cas nous n'avons pas besoin de résoudre un problème (NP-complet) d'ALB classique, puisque le temps de cycle total est implicitement minimisé.

La donnée initiale est *le taux de production imposé*. Nous savons que le taux de production peut être estimé en fonction du nombre d'ouvriers, n, et du poste de travail le plus chargé. Par conséquence, le but de la conception est de trouver la meilleure combinaison entre n et la configuration des m postes de travail, d'une part, et le chargement de chaque poste, $\{p_k\}_{k=1,2,...m}$, de l'autre part, afin d'obtenir le taux de production imposé.

Dans [BART 95] on trouve deux résultats importants à utiliser dans l'étape de la conception d'une ligne de type "bucket brigade" (ou TSS). Ils s'appuient sur l'assomption supplémentaire que tous les ouvriers sont identiques. Ce qui se traduit par le fait que leurs vitesses de travail sont supposées égales et égales à la vitesse standard : $v_1=v_2=...=v_n$. Les ouvriers sont dits "*ouvriers standards*". Les deux résultats sont listés ci-après.

THÉORÈME T-V.6 :

Une ligne TSS avec n ouvriers standards atteint potentiellement un taux de production maximum de $min\left\{n, \frac{1}{p_{max}}\right\}$ articles dans l'unité de temps.

Remarque :

Ce cas ne correspond pas aux conditions du théorème T-V.3, qui montre que "le bon rangement" de *n ouvriers distincts* garantit la convergence des positions de réinitialisation, et, comme conséquence, la convergence du taux de production.

Dans ce cas-là *seulement* le taux de production converge et atteint effectivement son maximum, même s'il existe des blocages, alors que les quantités de travail exécutées par chaque ouvrier (les positions de réinitialisation) **ne convergent pas**.

Conséquence :

Le taux de production d'une ligne TSS avec n ouvriers standards est indépendant de l'ordonnancement des postes de travail.

Dans le travail cité ci-dessus, l'analyse de plusieurs cas pratiques a conduit à quelques *règles de conception*, plutôt empiriques, mais utiles pour toute implantation pratique :

- le nombre d'ouvriers doit être moindre, mais proche de $1/p_{max}$; par exemple, dans l'industrie du prêt-à-porter, il est recommandable que $m \approx 2.5 \cdot n$;
- une équipe moins nombreuse se comporte mieux – en pratique, le plus souvent $n \in \{3,...6\}$;
- dans les cas où il existe des variations importantes des temps opératoires, les équipes plus nombreuses sont préférables, lorsqu'elles réduisent la probabilité des blocages.

Comme conclusion, nous listons ci-dessous **les principaux avantages** des lignes TSS :

- la ligne est gouvernée par une règle de fonctionnement – *la règle TSS* – simple et identique pour tous les ouvriers ;
- il n'y a *pas d'accumulations d'articles* sur la ligne, lorsque l'entrée d'un nouveau article est commandée par la sortie du précédent (une ligne TSS illustre le concept de "*pull system*", aussi bien *globalement*, que *localement*, au niveau de chaque ouvrier) ;

- *le transport* et *la manipulation* des articles sont aisés : les ouvriers transportent eux-mêmes les articles à travers la ligne ;
- grâce à la *propriété d'auto-équilibrage*, il n'est *pas nécessaire de mesurer exactement* les durées des tâches ; de plus, la ligne répond *spontanément* et *instantanément* aux changements de configuration, grâce à sa *capacité d'auto-organisation* ;
- *le taux de production* peut être *facilement ajusté*, simplement en ajoutant ou en enlevant des ouvriers sur/de la ligne.

Les lignes de fabrication de type TSS sont implantées notamment dans l'industrie du prêt-à-porter et dans la distribution des marchandises. En général, les lignes TSS sont applicables surtout dans les industries où :

- le travail requiert presque *la même habileté* pour toutes les opérations d'une ligne de fabrication ;
- chaque ouvrier peut *aisément bouger* parmi les postes de travail ;
- les postes sont beaucoup moins chers que la main d'œuvre ;
- il existe des *variations importantes de la demande* des produits fabriqués.

Le principe de fonctionnement des lignes auto-équilibrées appliqué aux lignes d'assemblage assure que le pilotage soit englobé dans le système lui-même. Il n'est pas donc nécessaire de concevoir un nouveau système de pilotage, comme dans les approches classiques. De ce point de vue, nous pouvons dire que telles lignes illustrent le concept de *fabrication "souple"* (*"lean" manufacturing* en anglais).

V.3 Analyse des systèmes d'assemblage auto-équilibrés comme systèmes dynamiques hybrides

V.3.1 Nécessité de l'approche

L'analyse par simulation des lignes d'assemblage auto-équilibrées montre une certaine ressemblance de celles-ci avec les systèmes dynamiques non linéaires, puisque nous avons mis en évidence les situations de comportement de type cycle limite. En fait, c'est justement cette analogie qui nous a permis de construire un *modèle non linéaire* d'une ligne de ce type, puis un schéma de simulation utilisant des blocs fonctionnels non linéaires.

Si nous étendons la définition de la trajectoire d'état pendant la durée des itérations, nous remarquons sa ***discontinuité***. Cela est une conséquence d'une hypothèse du modèle normatif : *la durée nulle du déplacement en arrière* (vitesse infinie), ou, autrement dit, *la réinitialisation simultanée et instantanée* de tous les ouvriers, qui détermine ***le changement brusque*** de leurs états initiaux. Dans un premier instant, cela a été une simplification nécessaire pour comprendre l'essentiel du fonctionnement. D'une part, renoncer à cette hypothèse rendrait inutilement compliqué le modèle de la ligne. D'autre part, cette hypothèse entraîne que l'approche de modélisation non linéaire ne soit pas suffisante, lorsqu'une telle approche demande la *continuité* des trajectoires d'état.

Nous remarquons, donc, que le comportement d'une ligne auto-équilibrée est linéaire et continu entre l'apparition de certains événements discrets. Ce qui signifie que *la dynamique de la ligne est aussi bien continue, que due à d'événements discrets*. Cette description correspond aux ***systèmes dynamiques hybrides***.

Par la suite nous maintenons les hypothèses du modèle normatif (voir V.2.2). En effet, les ouvriers sont essentiellement des intégrateurs de leurs vitesses de travail, donc leur comportement est *continu* et *linéaire* tant qu'ils ne sont pas forcés de réagir à deux types d'événements discrets : *le blocage* et *la réinitialisation*. À l'égard de ces deux situations, nous pouvons respectivement identifier **deux phénomènes hybrides** ([BRAN 98], [FLAU 98]) :

- *la commutation autonome*
- et *le saut autonome*.

Chaque ouvrier peut avoir *deux états* pendant une itération : soit il avance à sa vitesse de travail, soit il est bloqué à la frontière d'un poste occupé (sa vitesse devient nulle). **La commutation autonome** caractérise la transition entre ces deux états, qui sont *les états discrets* des ouvriers. Dans ce cas, la trajectoire d'état reste continue. Elle devient discontinue seulement dans les moments de réinitialisation, lorsque le dernier ouvrier atteint la fin de la ligne. Cela représente **le saut autonome**.

Les remarques ci-dessus nous suggèrent que les lignes d'assemblage auto-équilibrées peuvent être modélisées en tant que systèmes dynamiques hybrides. Ce type de système dynamique est caractérisé aussi bien par des *variables continues*, que par des *variables discrètes*. Il est décrit par des équations qui dépendent en général sur les deux types de variables. L'interaction des deux dynamiques, continue et discrète, se produit lorsque l'état continu franchit certaines zones dans l'espace d'états continus.

La théorie des systèmes dynamiques hybrides nous permet une nouvelle démarche de modélisation des lignes d'assemblage auto-équilibrées, mais aussi bien une analyse de stabilité. En ce qui suit nous allons encadrer ces lignes d'assemblage dans la classe des ***systèmes dynamiques hybrides à commutations et sauts autonomes***, selon la taxonomie proposée dans [BRAN 98]. Puis, nous allons montrer que la propriété d'auto-équilibrage est liée à la notion de "stabilité". Dans ce but, nous utiliserons le formalisme des ***systèmes dynamiques hybrides à mouvements discontinus*** ([YE 95]), où on définit des concepts semblables à ceux de la théorie classique des systèmes dynamiques. Finalement, nous allons donner *une autre démonstration* de la condition suffisante de comportement périodique optimal, en utilisant les ***critères de stabilité des systèmes dynamiques discrets***.

V.3.2 Modélisation des lignes d'assemblage auto-équilibrées

Deux approches des systèmes dynamiques hybrides peuvent être actuellement remarquées [BRAN 98] :

- l'approche par *agrégation* et
- l'approche par *continuation*.

Les deux approches sont duales en ce qui concerne l'accent donné à chacun des deux aspects des systèmes dynamiques hybrides, discret et continu.

Ainsi, l'approche par agrégation tend à traiter le système entier comme un automate fini ou comme *un système dynamique à événements discrets*, usuellement par la partition de l'espace d'états continus en cellules. Au contraire, la deuxième approche tend à modéliser le système par *une seule équation différentielle*. Ce qui se fait en englobant les actions discrètes au sein des équations différentielles non linéaires, ou bien en traitant les actions discrètes en tant que perturbations de certaines – habituellement linéaires – équations différentielles. Remarquons que nous avons déjà utilisé cette approche lors de l'analyse par simulation.

Dans le travail cité, la construction d'une taxonomie des systèmes dynamiques hybrides commence avec l'observation générale que la dynamique d'un système dynamique hybride est décrite par une équation différentielle :

$$\dot{x}(t) = \xi(t), \quad t \geq 0,$$

où $x(t)$ est la composante continue de l'état du système, à des valeurs dans un sous-ensemble de l'espace euclidien, et $\xi(t)$ est un champ vectoriel, généralement dépendant de l'état continu, $x(t)$, de la commande continue, $u(t)$, et de certains *phénomènes discrets*.

L'étude des exemples rencontrés dans le monde réel suggère que tout modèle d'un système dynamique hybride doit permettre la représentation de quatre types de *phénomènes hybrides* :

- la commutation autonome,
- le saut autonome,
- la commutation commandée
- et le saut commandé.

D'où vient la classification de systèmes dynamiques hybrides en deux grandes classes – *autonomes* et *commandés* – chacun d'eux pouvant être soit *à commutations*, soit *à sauts*, soit *à commutations et sauts*. Nous pouvons encadrer une ligne d'assemblage auto-équilibrée dans la classe des **systèmes dynamiques hybrides à commutations et sauts autonomes**. Ce modèle s'obtient simplement comme *combinaison* des modèles à commutations autonomes, respectivement à sauts autonomes.

Le modèle général d'un *système dynamique hybride à commutations autonomes* est :

$$\text{(V.8)} \qquad \begin{cases} \dot{\underline{x}}(t) = h(\underline{x}(t), \underline{q_w}(t)) \\ \underline{q_w}^+(t) = \nu(\underline{x}(t), \underline{q_w}(t), \underline{q_s}(t)) \\ \underline{q_s}^+(t) = \mu(\underline{x}(t), \underline{q_s}(t)) \end{cases}$$

Dans le cas d'une ligne auto-équilibrée avec n ouvriers et m postes de travail, les éléments du modèle sont :

$\underline{x}(t) = [x_1(t) \;\; x_2(t) \;\; \ldots \;\; x_n(t)]^T \in \mathbf{R}^n (\in [0;1]^n)$ – *le vecteur d'état continu* (les positions instantanées des ouvriers)

$\underline{q_w}(t) = [q_{w_1}(t) \;\; q_{w_2}(t) \;\; \ldots \;\; q_{w_n}(t)]^T \in \{0,1\}^{n-1} \times \{1\}$ – *le vecteur d'état discret des ouvriers*, $\begin{cases} 0 - \text{blocage} \\ 1 - \text{avancement} \end{cases}$

$\underline{q_s}(t) = [q_{s_1}(t) \;\; q_{s_2}(t) \;\; \ldots \;\; q_{s_m}(t)]^T \in \{0,1\}^m$ – *le vecteur d'état discret des postes*, $\begin{cases} 0 - \text{libre} \\ 1 - \text{ocupé} \end{cases}$

$(\cdot)^+$ – l'état suivant à la commutation

Rappelons la **notation** :

$$P_j = \sum_{k=0}^{j} p_k, \; j=\overline{1,m}, \quad P_0 = 0, \; P_m = 1,$$

qui désigne les quantités de travail *cumulées* jusqu'au poste j, y compris, $j=1,2,\ldots m$.

Nous utiliserons aussi la **notation** :

$$\underline{v} = [v_1 \;\; v_2 \;\; \ldots \;\; v_n]^T$$

Les fonctions de la relation (V.8) résultent de manière implicite :

$\Rightarrow$ fonction h :

$$(V.9) \qquad \underline{\dot{x}}(t)=\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \cdot\\ \cdot\\ \cdot\\ \dot{x}_n(t)\end{bmatrix}=\underbrace{\begin{bmatrix}q_{w_1}(t) & & & & \\ & q_{w_2}(t) & & 0 & \\ & & \cdot & & \\ & & & \cdot & \\ & 0 & & & \\ & & & & q_{w_n}(t)\end{bmatrix}}_{\overset{notation}{=}\ A_q=diag\{\underline{q_w}(t)\}}\cdot\underline{v}$$

$\Rightarrow$ fonction ν :

$$q^+_{w_i}(t)=\begin{cases}0, q_{w_i}(t)=1, \exists j:(x_i(t)=P_{j-1})\wedge(q_{s_j}=1)\\ 1, q_{w_i}(t)=0, \exists j:(x_i(t)=P_{j-1})\wedge(q_{s_j}=0), \qquad i=\overline{1,n-1}\\ q_{w_i}(t), \text{autrement}\end{cases}$$

$$q^+_{w_n}(t)=1$$

$\Rightarrow$ fonction μ :

$$q^+_{s_j}(t)=\begin{cases}0, q_{s_j}(t)=1, \exists i: x_i(t)=P_j\\ 1, q_{s_i}(t)=0, \exists i: x_i(t)=P_{j-1}, \qquad j=\overline{1,m}\\ q_{s_j}(t), \text{autrement}\end{cases}$$

$$q^+_{w_n}(t)=1$$

Le modèle général d'un *système dynamique hybride à sauts autonomes* est décrit par :

$$(V.10) \qquad \begin{cases}\underline{\dot{x}}(t)=h(\underline{x}(t),\underline{q_w}(t)), & \underline{x}(t)\notin M\\ \underline{x}^+(t)=\mathrm{J}(\underline{x}(t)), & \underline{x}(t)\in M\end{cases}$$

Dans notre cas, l'ensemble $M=\{m_1, m_2, \ldots, m_{n-1}, 1\} \in [0,1]^{n-1}\times\{1\}$ (les sauts décrivent la réinitialisation de la ligne, c'est à dire, le moment où $x_n(t)=1$).

Dans la relation (V.10), la fonction h est donnée par (V.9) et la fonction J est détaillée par la suite :

$$(V.11) \qquad \mathrm{J}(\underline{x}(t))=\underbrace{\begin{bmatrix}0 & 0 & \ldots & 0 & 0\\ 1 & 0 & 0 & \ldots & 0\\ 0 & 1 & \ldots & \ldots & \ldots\\ \ldots & \ldots & \ldots & 0 & 0\\ 0 & \ldots & 0 & 1 & 0\end{bmatrix}}_{P}\cdot\underline{x}(t)$$

Le modèle d'une ligne avec auto-équilibrage vue comme système dynamique hybride à commutations et sauts autonomes est obtenu lors du couplage des deux modèles ci-dessus (relations (V.8) et (V.10)) :

$$\text{(V.12)}\quad \begin{cases} \dot{\underline{x}}(t) = h(\underline{x}(t), \underline{q_w}(t)) = A_q \cdot \underline{v}, & \underline{x}(t) \notin M \quad (x_n(t) \neq 1) \\ \underline{x}^+(t) = J(\underline{x}(t)) = P \cdot \underline{x}(t), & \underline{x}(t) \in M \quad (x_n(t) = 1) \\ \underline{q_w}^+(t) = \nu(\underline{x}(t), \underline{q_w}(t), \underline{q_s}(t)) \\ \underline{q_s}^+(t) = \mu(\underline{x}(t), \underline{q_s}(t)) \end{cases},$$

avec les conditions initiales : $0 \leq x_{1_0} < x_{2_0} < \ldots < x_{n_0} < 1$.

V.3.3 Étude de la stabilité

L'analyse de la stabilité est un point essentiel de toute démarche d'analyse d'un système dynamique. Pratiquement, le but de cette analyse est de *caractériser l'ensemble de toutes les trajectoires possibles sans les calculer explicitement*. Ce but est poursuivi aussi bien en automatique, qu'en informatique, mais évidemment en utilisant des approches différentes. Ainsi, ce qu'on appelle "*analyse de stabilité*" dans les approches systémiques est connu sous le nom de "*validation du modèle symbolique*" en informatique ([FLAU 98]). Les deux approches sont complémentaires.

Dans [YE 95] on présente une approche unitaire des systèmes dynamiques hybrides, appropriée à la formulation du problème de l'analyse de stabilité. Il s'agit du formalisme des *systèmes dynamiques hybrides à mouvements (ou trajectoires) discontinu(e)s*. Les définitions des concepts bien connus dans la théorie classique des systèmes dynamiques – comme, par exemple, *espace de temps*, *mouvement*, *ensemble invariant*, *point d'équilibre* – sont adaptées aux caractéristiques des systèmes dynamiques hybrides. Nous avons listé les éléments principaux de ce formalisme en Annexe C.

Dans ce qui suit, nous allons montrer que le comportement des lignes d'assemblage avec auto-équilibrage peut être analysé en termes du formalisme susmentionné.

Une ligne d'assemblage auto-équilibrée avec n ouvriers peut être décrite comme un 5-uple $\{T, X, A, S, T_0\}$ – qui est un système dynamique hybride (voir Annexe C) – par la suite :

$T=\mathbf{R}^+$ – le système évolue continûment en temps ;

$T_0 = \{t_0 | t_0 \in T\}$ – l'ensemble des moments initiaux ;

$X=\mathbf{R}^n$ – l'ensemble d'états ;

$X \supset A = \{a | a = [x_{1_0} \; x_{2_0} \ldots x_{n_0}]^T\}$ – l'ensemble des états initiaux;

$$S \subset \{p(t,a,t_0) = \underline{x}(t,a,t_0) = [x_1(t,a,t_0) \quad x_2(t,a,t_0) \quad \ldots \quad x_n(t,a,t_0)]^T\},$$

où $p(t_0, a, t_0) = a$.

Remarquons que, dans l'hypothèse des vitesses constantes et du bon ordre des ouvriers – $v_1 < v_2 < \ldots < v_n$ – qui, en plus, ne sont jamais bloqués, les mouvements p – c'est à dire les trajectoires – sont les solutions de l'équation différentielle suivante :

$$\text{(V.13)} \quad \dot{\underline{x}}(t,a,t_0) = [v_1\ v_2 \ldots\ v_n]^T,$$

avec les conditions initiales : $0 \le x_{1_0} < x_{2_0} < \ldots < x_{n_0} < 1$.

La formulation du problème d'analyse de la stabilité d'un système dynamique hybride défini comme ci-dessus nécessite l'introduction du concept de "*stabilité des ensembles invariants*", ou simplement de "*stabilité des invariants*" (voir Annexe C). Un *invariant* est un sous-ensemble d'états initiaux caractérisé par le fait que tout mouvement partant d'un état contenu dans ce sous-ensemble reste à l'intérieur de celui-ci. Un *point d'équilibre* est un invariant formé par un seul état. Ce qui signifie que l'équilibre, une fois atteint, ne sera jamais quitté par aucun mouvement.

Les règles de fonctionnement d'une ligne d'assemblage de type TSS – plus précisément, la règle de la réinitialisation de la ligne et la règle de priorité sur un poste de travail– nous permettent de formuler la proposition ci-dessous.

Proposition P-V.1

Si le système dynamique décrit par la relation (V.13) est réinitialisé chaque fois que $x_n(t)=1$ conformément à :

$$\text{(V.14)} \quad \begin{cases} x_1(t) = 0 \\ x_i(t) = x_{i-1}(t^-), \quad i = \overline{2,n} \end{cases}$$

alors le sous-ensemble $[0,1]^n$ de l'espace d'états est **un invariant** du système (la notation "t^-" signifie le moment immédiatement avant t).

Justification :

Tout d'abord, remarquons que la relation (V.13), lorsqu'elle est assignée au fonctionnement d'une ligne TSS, suppose implicitement l'hypothèse des *vitesses de travail constantes* et l'hypothèse du *bon ordre*. Donc $v_1 < v_2 < \ldots < v_n$.

Soit a de $[0,1]^n$ l'état initial et soit t_0 l'instant initial. Nous pouvons considérer que $t_0=0$ sans perte de généralité. Nous devons prouver que tous les mouvements décrits par $\underline{x}(t,a,t_0)$ restent à l'intérieur de $[0,1]^n$.

Soit r_1 le premier moment où $x_n(t)=1$, c'est à dire la fin de la première itération, ou, autrement dit, le premier moment de réinitialisation. Tout en gardant la signification des notations, nous pouvons écrire :

$$\underline{x}(0,a,0) = a = [x_{1_0}\ x_{2_0} \ldots x_{n_0}]^T = \underline{x}^{(0)} \in [0,1]^n$$

$$\underline{x}(r_1,a,0) = [x_1(r_1)\ x_2(r_1) \ \ldots\ x_n(r_1)]^T = \underline{x}^{(1)}$$

Par la suite nous abandonnons les notations désignant l'état initial et le moment initial. Notons $\underline{y}^{(0)} = \underline{x}(r_1^-)$. De la relation (V.14) on tire que :

$$\text{(V.15)} \quad \left.\begin{cases} x_1(r_1) = 0 \\ x_i(r_1) = x_{i-1}(r_1^-), \qquad i = \overline{2,n} \end{cases}\right\} \Leftrightarrow \underline{x}^{(1)} = P \cdot \underline{y}^{(0)},$$

où la matrice P modélise le saut autonome de réinitialisation (voir (V.11)). Conformément à la relation (V.13) et aux hypothèses implicitement maintenues, il vient que :

$$\begin{cases} y_i^{(0)} = x_i^{(0)} + v_i \cdot r_1 < x_{i+1}^{(0)} + v_{i+1} \cdot r_1 = y_{i+1}^{(0)}, & i = \overline{1,n-1} \\ y_n^{(0)} = 1 \end{cases}$$

Donc, $x_1^{(0)} \le y_1^{(0)} < y_2^{(0)} < \ldots < y_{n-1}^{(0)} < y_n^{(0)} = 1$, ce qui implique $\underline{y}^{(0)} \in [0,1]^n$. Évidemment, les positions des ouvriers pendant la première itération respectent $\underline{x}^{(0)} \le \underline{x}(t) < \underline{y}^{(0)}$. Il en résulte que $\underline{x}(t) \in [0,1]^n, \forall t \in [0, r_1)$ et, en utilisant (V.15), que $\underline{x}^{(1)} \in [0,1]^n$.

Le même raisonnement peut être utilisé afin de prouver par récurrence selon k le fait que $\underline{x}(t) \in [0,1]^n, \forall t \in [r_{k-1}, r_k), k = 1,2,\ldots$. Remarquons que $\underline{x}^{(k)}$ peut être regardé comme l'état initial de l'itération k. La justification est ainsi finie.

Remarque :

Lorsque l'ensemble $[0,1]^n$ est un invariant d'une ligne avec vitesses constantes, en bon ordre et sans blocages, cet ensemble est tant plus un invariant d'une ligne quelconque. Pratiquement, ce fait est une conséquence de la règle de priorité de l'ouvrier d'indice supérieur sur un poste de travail.

Conformément au théorème T-V.3, une ligne qui respectent les conditions susmentionnées possède comme point d'équilibrage (point fixe) :

$$\text{(V.16)} \quad \underline{x}^* = \left[\; 0 \quad \frac{v_1}{\sum_{j=1}^{n} v_j} \quad \ldots \quad \frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j} \quad \ldots \quad \frac{\sum_{j=1}^{n-1} v_j}{\sum_{j=1}^{n} v_j} \; \right].$$

Dans un premier temps, lors de la modélisation d'une telle ligne d'assemblage en tant que système dynamique hybride, nous pourrions penser que le point fixe, donné par l'expression (V.16), peut être caractérisé comme un (point d') équilibre au sens des définitions C.4 et C.5 (voir Annexe C). Cela n'est pas vrai, puisque, en fait, les mouvements (les trajectoires) ne convergent pas vers un seul point dans l'espace d'états – ce qui signifierait que les ouvriers arrêtent – mais vers un *schéma* de mouvement, où chaque ouvrier *i répète* à chaque itération l'exécution du même contenu de travail, dans l'intervalle $\left[\frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}, \frac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j} \right]$.

En général, si la ligne atteint un comportement stabilisé, celui-ci est périodique (voir aussi V.2.4, l'analyse par simulation).

V.3.4 Convergence vers le comportement périodique – approche par la théorie des systèmes dynamiques discrets

Ce sous-paragraphe présente une nouvelle démonstration du théorème T-V.3, que nous avons énoncé dans V.2.3.1. Ce théorème a été premièrement prouvé dans un contexte *stochastique* ([BART 96a], repris en Annexe A). Rappelons qu'il indique une *condition suffisante* devant être remplie par la configuration d'une ligne TSS pour atteindre *l'auto-équilibrage*, c'est à dire *le comportement stabilisé caractérisé par le temps de cycle – ou par le taux de production – optimal*.

Par rapport à la démonstration originale, la notre se place dans un contexte *déterministe*, en faisant appel aux critères de stabilité des systèmes dynamiques discrets. Nous gardons les notations.

Reprenons l'énoncé du théorème :

THÉORÈME T-V.3 :

Si les vitesses sont *constantes*, $v_i(x)=v_i$, avec $v_1<v_2<....<v_n$, et si, de plus, les ouvriers ne sont *jamais bloqués*, la ligne converge vers *l'unique point fixe* $\underline{x}^*$, donné par l'expression (V.16). Le taux de production est $\sum_{j=1}^{n} v_j$, le meilleur possible.

Démonstration :

Nous avons à montrer la convergence des positions de réinitialisation. Ce qui se réduit à la convergence de la suite $\left\{\underline{x}^{(k)}\right\}_{k=0,1,...}$.

Dans les conditions du théorème, les positions de réinitialisation évoluent selon l'équation récurrente suivante :

(V.17) $\quad \underline{x}^{(k+1)} = A \cdot \underline{x}^{(k)} + \underline{v}^*, \quad k = 0,1,...$

où : $\quad A = \begin{bmatrix} 0 & 0 & ... & 0 & 0 \\ 1 & 0 & ... & 0 & -\frac{v_1}{v_n} \\ 0 & 1 & 0 & ... & -\frac{v_2}{v_n} \\ & & ... & & \\ 0 & ... & 1 & 0 & -\frac{v_{n-2}}{v_n} \\ 0 & ... & 0 & 1 & -\frac{v_{n-1}}{v_n} \end{bmatrix},$

$$\underline{v}^* = \begin{bmatrix} 0 & \frac{v_1}{v_n} & ... & \frac{v_{n-2}}{v_n} & \frac{v_{n-1}}{v_n} \end{bmatrix}^T.$$

La relation (V.17) décrit la dynamique d'un système dynamique linéaire discret invariant, ayant $\left\{\underline{x}^{(k)}\right\}_{k=0,1,...}$ pour trajectoire d'état. Remarquons que la matrice A est sous forme canonique compagne observable.

Premièrement, remarquons que le point $\underline{x}^*$ vérifie l'équation (V.17). Notons $\underline{y}^{(k)} = \underline{x}^{(k)} - \underline{x}^*$. Il s'ensuit que :

$$\underline{y}^{(k+1)} = \underline{x}^{(k+1)} - \underline{x}^* = A \cdot (\underline{x}^{(k)} - \underline{x}^*) = A \cdot \underline{y}^{(k)}, \quad t = 0,1,...$$

Par conséquent, c'est la convergence de la suite $\left\{\underline{y}^{(k)}\right\}_{k=0,1,...}$ qui doit être démontrée.

Cette suite représente la trajectoire d'état d'un système dynamique linéaire discret invariant, dont l'équation d'état est :

$$\underline{y}^{(k+1)} = A \cdot \underline{y}^{(k)}, k = 0,1,...$$

La *convergence de la trajectoire d'état* de ce système est équivalente à sa *stabilité*.

Par la suite, nous allons utiliser quelques résultats théoriques connus (voir Annexe D).

Rappelons qu'*une condition nécessaire et suffisante* pour la stabilité d'un système dynamique discret est que sa *matrice d'état soit Schur-stable*, ce qui signifie que toutes ses valeurs propres doivent se trouver strictement dans le cercle unité. Dans ce cas, cela s'écrit :

$$\left|\lambda_i(A)\right| < 1, i = \overline{1,n}.$$

Donc, la stabilité de la matrice A s'exprime par la *convergence de son polynôme caractéristique*. Dans notre cas, ce polynôme est :

$$\Delta(\lambda) = \lambda^n + \frac{v_{n-1}}{v_n} \cdot \lambda^{n-1} + ... + \frac{v_1}{v_n} \cdot \lambda .$$

Le critère de Kakeya (voir Annexe D) fournit *une condition suffisante* pour la convergence d'un polynôme général de la forme

$$\Delta(\lambda) = a_n \cdot \lambda^n + a_{n-1} \cdot \lambda^{n-1} + ... + a_1 \cdot \lambda + a_0, \qquad a_i \in \mathbf{R},\ i = \overline{0,n},\ a_n > 0,$$

à savoir : $a_n > a_{n-1} > ... > a_1 > a_0 > 0$.

Dans notre cas, de l'utilisation de ce critère il résulte qu'il est suffisant d'avoir $v_n > v_{n-1} > ... > v_1 (> 0)$, pour que la suite $\left\{\underline{y}^{(k)}\right\}_{k=0,1,...}$ converge. Il vient que :

$$\left\{\underline{y}^{(k)}\right\}_{k=0,1,...} \to 0 \Leftrightarrow \left\{\underline{x}^{(k)}\right\}_{k=0,1,...} \to \underline{x}^*,$$

où $\underline{x}^*$ est *le point fixe* de la ligne, obtenu comme :

$$\underline{x}^* = (I_n - A)^{-1} \cdot \underline{v}^*,$$

ce qui est équivalent à l'expression (V.16).

Donc, la condition $v_n > v_{n-1} > ... > v_1$, qui **exprime** ***le rangement des ouvriers du plus lent au plus rapide***, est *suffisante* pour la convergence des positions de réinitialisation, $\left\{\underline{x}^{(k)}\right\}_{k=0,1,...}$, vers le point $\underline{x}^*$, qui caractérise *un comportement à un temps de cycle constant*. Cela résulte du fait que chaque ouvrier *i* exécute le même contenu de travail, situé entre $\frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}$ et $\frac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j}$, à chaque itération. Par hypothèse, aucun ouvrier n'est bloqué. Donc, la durée de chaque itération est temps de travail effectif. Elle représente *le temps de cycle* de la ligne, T_c, qui s'obtient par la suite :

$$T_c = \frac{\frac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j} - \frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}}{v_i} = \frac{1}{v_i} \cdot \frac{v_i}{\sum_{j=1}^{i} v_j} = \frac{1}{\sum_{j=1}^{i} v_j}.$$

Le taux de production, défini comme l'inverse du temps de cycle, est $\sum_{j=1}^{n} v_j$.

Tout autre placement des ouvriers sur la ligne contiendrait éventuellement des retards à cause de blocages, ce qui conduirait à une diminution de la productivité. Ainsi, le point fixe $\underline{x}^*$ – dont l'*unicité* est garantie par le lemme L-V.1 – caractérise ***le comportement optimal***, ce qui signifie qu'il représente *le point d'équilibrage* de la ligne.

Remarques :

1) En restant dans le domaine d'application de la théorie des systèmes dynamiques discrets, nous pouvons utiliser aussi d'*autres critères de convergence* d'un polynôme général :

$$\Delta(\lambda) = a_n \cdot \lambda^n + a_{n-1} \cdot \lambda^{n-1} + ... + a_1 \cdot \lambda + a_0, \qquad a_i \in \mathbf{R},\ i = \overline{0,n},\ a_n > 0 .$$

Par exemple, une ***condition nécessaire*** est :

$$\begin{cases} \Delta(1) > 0, \\ (-1)^n \cdot \Delta(-1) > 0, \\ a_n > |\, a_0 \,|, \end{cases}$$

ce qui conduit dans notre cas à :

$$\begin{cases} n = 2 \cdot k, & \sum_{j=1}^{k} v_{2 \cdot j} > \sum_{j=1}^{k} v_{2 \cdot j-1} \\ n = 2 \cdot k + 1, & \sum_{j=1}^{k+1} v_{2 \cdot j-1} > \sum_{j=1}^{k} v_{2 \cdot j} \end{cases} .$$

Remarquons que cette condition exprime toujours une contrainte technologique de placement des ouvriers sur la ligne, mais plus compliquée et plus difficile à mettre en œuvre que celle issue du théorème ci-dessus. Cette dernière condition est *nécessaire* et nous pouvons remarquer qu'elle est implicitement remplie lors du remplissage de la condition de "bon ordre", qui est *suffisante*.

2) Le cas où tous les coefficients du polynôme $\Delta(\lambda)$ sont ***égaux*** correspond à un système dynamique discret ***critiquement stable***.

Dans le cas d'une ligne TSS, l'égalité des coefficients polynomiaux conduit à l'*égalité des vitesses de travail* : $v_1=v_2=...=v_n$.

D'autre part, la stabilité critique est généralement caractérisée par une réponse temporelle bornée, mais pas convergente. Ce qui se traduit dans notre cas par des *positions de réinitialisation qui ne convergent pas*, mais qui se répètent après un certain nombre d'itérations. Nous avons analysé par simulation quelques situations de ce type, où nous avons mis en évidence ***les cycles limites*** dans l'espace d'états (voir V.2.4.2).

V.4 Conclusion

Ce chapitre a été dédié à l'étude d'une classe spéciale de systèmes d'assemblage, qui possèdent la *capacité d'auto-organisation*. Leur principe de fonctionnement est applicable non seulement dans le cas de l'assemblage, mais en général aux systèmes de production. Ces systèmes sont connus sous le nom de "*bucket brigades*". Au cours de ce chapitre, nous les avons appelés "**lignes TSS**", selon leur première implantation pratique, comme "Toyota Sewn Management System".

Nous avons vu que les particularités des lignes TSS par rapport aux lignes d'assemblage classiques reposent sur le fait qu'elles utilisent des *opérateurs humains*, dont la dynamique est gouvernée par une règle simple – **la règle TSS**. Celle-ci permet **l'auto-équilibrage spontané** de la ligne, ce qui représente un grand avantage du point de vue de la conception. Par suite, il n'est pas nécessaire de résoudre un problème d'ALB classique. Nous avons énuméré les avantages de ce type de ligne, par exemple, le transport facile, l'ajustement simple du taux de production, etc. Globalement, les lignes TSS illustrent le concept de la *fabrication* "*souple*", lorsque le pilotage est englobé dans le système lui-même.

La modélisation des lignes avec auto-équilibrage a eu comme point de départ les travaux de deux chercheurs américains, J.J. Bartholdi et D.D. Eisenstein, qui ont obtenu des résultats théoriques valorisants. Nous avons présenté ces résultats, dont les plus importants sont accompagnés par des démonstrations assez élaborées (voir Annexe A).

Cette première approche est construite autour de quelques hypothèses simplificatrices, réunies sous le nom de "**modèle normatif**". Le point essentiel de ce modèle est la modélisation des ouvriers par **fonctions de vitesse de travail**. Les ouvriers peuvent bouger entre postes de travail successives pour continuer le travail sur un article (une instance du produit à assembler), en respectant la priorité de l'ouvrier plus avancé. La règle TSS leur impose un *comportement périodique*, repris chaque fois quand un produit fini sort de la ligne. Les auteurs cités ont montré que l'évolution d'une ligne TSS peut être décrite comme **processus stochastique**, par l'intermédiaire d'*une chaîne de Markov*.

En ce cadre formel, le plus important des résultats théoriques est ***une condition suffisante*** pour obtenir un *comportement périodique stabilisé* – c'est à dire, avec un temps de cycle constant – qui est de plus ***optimal***. Une telle situation illustre la propriété d'***auto-équilibrage***. Cette condition s'exprime comme une contrainte technologique simple de placement des ouvriers sur la ligne : **du plus lent au plus rapide**, c'est à dire en ordre croissant de leur habileté de travail (théorème T-V.2). Si, de plus, la vitesse de travail de chaque ouvrier est *constante*, alors le point d'équilibrage et le taux de production – qui est optimal – peuvent être exactement calculés (théorème T-V.3).

Les **contributions originales** de ce chapitre regardent :

- l'analyse par simulation des lignes TSS sur *un schéma non linéaire* ;
- la modélisation des lignes TSS comme *systèmes dynamiques hybrides* et l'*étude de stabilité* ;
- *une nouvelle démonstration* du théorème T-V.3, en utilisant les critères de stabilité des systèmes dynamiques discrets.

À partir du modèle normatif, nous avons fait une analyse par simulation des lignes TSS, pour mieux comprendre ce que les auteurs susmentionnés ont appelé "comportement compliqué" et "comportement anomal". La mise en évidence des différents cas de figures a été faite en modélisant une ligne TSS à l'aide d'**un système d'équations différentielles non linéaires** (voir relation (V.7)). Ce qui nous a permis de construire en Simulink un schéma de simulation formé par blocs non linéaires (voir Annexe B).

Nous avons montré que, malheureusement, le modèle non linéaire obtenu de cette façon n'est pas suffisant lors d'une démarche d'analyse, à cause des ***discontinuités*** de la trajectoire d'état. C'est ainsi que nous avons justifié la nécessité de l'approche par la théorie des **systèmes dynamiques hybrides**. Nous avons identifié *deux phénomènes hybrides* dans la dynamique des lignes TSS – *la commutation autonome* et *le saut autonome* – dont le dernier détermine la rupture de la trajectoire d'état. Il en a résulté naturellement que les lignes avec auto-équilibrage peuvent être vues comme ***systèmes dynamiques hybrides à commutations et sauts autonomes***, selon la taxonomie proposée par M. S. Branicky. Nous avons présenté en détail la description mathématique de ce modèle.

L'existence du comportement stabilisé d'une ligne TSS peut être liée à la stabilité des systèmes dynamiques hybrides. Pour l'étude de stabilité nous avons utilisé le formalisme des ***systèmes dynamiques hybrides à mouvements discontinus***, dont les éléments sont listés en Annexe C. Cette approche nous a permis de mettre en évidence **un ensemble invariant** d'une ligne TSS vue comme système dynamique hybride.

En dernier, nous avons donné une autre démonstration, plus simple, de la condition suffisante qui conduit à l'équilibrage d'une ligne TSS, dans le cas particulier des *vitesses de travail constantes* (une nouvelle démonstration du théorème T-V.3). En utilisant **un critère de stabilité des systèmes dynamiques discrets** (le critère de Kakeya), nous avons montré que "le bon ordre" des ouvriers sur la ligne assure la convergence des positions de réinitialisation et, par conséquent, un taux de production constant, qui, de plus, est le meilleur possible.

Par les contributions de ce chapitre, nous avons essayé d'établir un cadre de développement d'une approche systémique des systèmes d'assemblage avec auto-équilibrage.

Conclusion générale

En raison de sa complexité, la conception des systèmes d'assemblage est un problème difficile, dont la résolution algorithmique n'est pas possible. D'autre part, on ne peut pas se contenter d'une solution basée uniquement sur l'expérience du concepteur. Celui-ci doit être guidé dans sa démarche par des résultats théoriques bien fondés. Sans avoir pour but d'élaborer une nouvelle méthodologie de conception, notre travail présenté ici s'est proposé de fournir des méthodes et des outils complémentaires aux approches de conception largement utilisées.

Le graphe de précédence est un outil puissant pour la conception des systèmes d'assemblage, étant le seul modèle des processus d'assemblage utilisé par les méthodes issues de l'équilibrage des lignes d'assemblage. Cela s'explique par deux qualités principales : la flexibilité et la compacité de la représentation.

Nous avons proposé une approche systématique de représentation d'un ensemble de gammes (séquences) d'assemblage par un ensemble de graphes de précédence. La résolution de ce problème a requis l'introduction de quelques concepts nouveaux. Elle permet l'exploitation de la précision du premier modèle des processus d'assemblage – pour l'obtention duquel on dispose de méthodes bien fondées – sur la structure équivalente, mais plus "souple", du deuxième, qui est utilisé par la plupart des méthodes de conception des systèmes d'assemblage.

Chaque gamme d'assemblage est un ordre total des symboles représentant les tâches d'assemblage. Le graphe de précédence exprime l'ordre chronologique partiel des tâches d'assemblage. Par l'intermédiaire de la classification et de la décomposition, nous avons montré que notre problème se réduit à la détermination des graphes de précédence linéaires qui correspondent à un ensemble de gammes d'assemblage linéaires contenant la même tâche de chargement.

Nous nous sommes occupés d'abord de l'équivalence d'un ensemble de gammes à un seul graphe de précédence.

Nous avons montré qu'un ensemble de gammes donné, Ω, conduit à un graphe de précédence, G, respecté par un ensemble de gammes, Θ, en général plus nombreux que Ω. L'égalité $\Omega=\Theta$ définit l'équivalence du graphe G avec l'ensemble de départ, Ω. Nous avons formulé et démontré une condition nécessaire et suffisante pour que cette équivalence soit remplie : un ensemble de gammes d'assemblage est représenté de façon biunivoque par un et un seul graphe de précédence si et seulement si il a une propriété notée Π (*théorème T-III.5*).

Pour la mise en évidence de cette propriété globale nous avons introduit la relation d'indifférence sur l'ensemble des symboles (tâches) – définie par complémentarité à la relation de précédence – et la notion de "gamme d'assemblage symétrique". La symétrique d'une gamme est définie par rapport à une succession directe de deux symboles indifférents : les deux gammes ont la même structure, sauf l'ordre des deux symboles. Par définition, un ensemble Ω de gammes possède la propriété Π si pour chaque gamme toutes ses symétriques appartiennent également à l'ensemble Ω.

Le point essentiel de la démonstration de la condition nécessaire et suffisante susmentionnée a été la déduction d'une forme unitaire d'écriture des gammes qui respectent un graphe de précédence. Cette forme résulte de l'organisation du graphe de la relation d'indifférence – le graphe d'indifférence – en composantes connexes (*théorème T-III.1* et *T-III.1 reformulé*).

Nous avons proposé que les résultats théoriques résumés ci-dessus soient exploités par deux algorithmes basés sur la génération de toutes les gammes symétriques à partir d'une gamme quelconque.

Nous avons proposé un premier algorithme, $A_1(\Omega)$ – *l'algorithme de décision*, qui détermine si un ensemble de gammes possède ou non la propriété Π.

Lorsqu'un ensemble de gammes ne possède pas la propriété Π, il n'est pas représenté par un seul graphe, mais par un ensemble équivalent de graphes de précédence. Nous avons proposé un deuxième algorithme, $A_2(\Omega)$ – *l'algorithme de partage*, qui fournit l'ensemble minimal de graphes de précédence équivalent à un ensemble de gammes (*proposition P-III.2*). L'idée de base de cet algorithme consiste à trouver les sous-ensembles maximaux qui ont la propriété Π. L'implémentation en Prolog des deux algorithmes est immédiate.

L'importance d'une méthode systématique d'obtention des graphes de précédence résulte de la formulation du problème de l'équilibrage des lignes d'assemblage, aussi bien dans le cas monoproduit, que dans le cas multiproduit. Le problème de l'équilibrage multiproduit peut être réduit au cas monoproduit si l'on dispose de graphes de précédence pour chaque produit de la famille. La signification du graphe de précédence dans le cas monoproduit peut être généralisée au cas multiproduit, quand il s'agit d'une famille iso-structurelle de produits.

Le problème de l'équilibrage est formulé en tant que problème d'affectation des tâches aux postes de travail, qui est optimale du point de vue du temps de cycle total du système. Nous avons passé en revue les deux aspects issus de l'équilibrage : l'obtention des découpages en postes et l'emploi des méthodes d'optimisation appropriées.

Conformément à la majorité des méthodes de conception des systèmes d'assemblage existantes, l'obtention d'un découpage porte sur le graphe de précédence et consiste à délimiter les sous-ensembles de tâches d'assemblage (nœuds du graphe de précédence) qui ont la structure de poste de travail ("postes candidats"). Nous avons montré que les composantes connexes du graphe d'indifférence ont la structure de poste de travail (*proposition P-IV.2*). Malheureusement, la réciproque n'est pas valable.

Le problème d'équilibrage est NP-complet, lorsqu'il est essentiellement un problème de partition optimale sur un graphe.

Nous avons proposé un autre point de vue, en présentant une formulation systémique du problème d'équilibrage en tant que problème d'optimisation discrète, adaptée à la résolution par programmation dynamique. Cette approche, même si elle ne présente pas d'intérêt pratique évident, a l'avantage de s'apparenter à un formalisme bien fondé théoriquement.

En connaissant la problématique de l'équilibrage des systèmes d'assemblage, nous avons présenté une étude de la classe des systèmes d'assemblage avec auto-équilibrage. Ces systèmes ont la capacité d'auto-organisation, en fonctionnant selon un principe applicable aux systèmes de production en général. Dans certaines conditions, ce principe leur permet de se comporter spontanément de manière optimale du point de vue de l'équilibrage, c'est à dire avec un temps de cycle minimum ou, équivalent, avec un taux de production maximal. Par conséquent, la conception de ces systèmes évite la résolution d'un problème d'ALB classique. Pour ces systèmes nous avons utilisé l'appellation de "lignes TSS", selon leur première implantation pratique, chez Toyota, dans les années '70.

En utilisant des opérateurs humains, qui peuvent bouger entre postes adjacents pour continuer à travailler sur une instance du produit à assembler, ces lignes sont la mise en œuvre du concept de "découpage dynamique" : lorsque sur chaque poste on exécute une seule tâche, la notion de "poste de travail", comprise comme ensemble de tâches, peut être attribuée à l'ensemble de postes successifs sur lesquels un ouvrier quelconque travaille au cours d'un cycle de fabrication ; les postes ainsi définis ont des frontières variables dans le temps.

Dans la littérature on a montré, sous les hypothèses du "modèle normatif" et dans un contexte *stochastique*, que la contrainte du placement des ouvriers du plus lent au plus rapide sur la ligne constitue une condition suffisante pour l'auto-équilibrage.

Nous avons essayé d'établir un cadre de développement d'une approche systémique originale des systèmes d'assemblage avec auto-équilibrage.

Nous avons proposé d'abord une analyse par simulation des lignes TSS, dans le cas des vitesses constantes de travail. Ainsi, en modélisant la ligne par un système d'équations différentielles non linéaires (*relation* V.7), nous avons construit en Simulink un schéma-bloc non linéaire (*Annexe* B). Les différents cas de figures ont mis en évidence les discontinuités de la trajectoire d'état et, donc, l'insuffisance de l'approche par la théorie des systèmes dynamiques non linéaires. L'approche d'analyse par la théorie des systèmes dynamiques hybrides – dont l'évolution est aussi bien continue, que due à d'événements discrets – s'est avérée ainsi nécessaire.

Nous avons mis en évidence deux phénomènes hybrides qui gouvernent la dynamique des lignes TSS : la commutation autonome et le saut autonome. Ce qui les encadre dans la classe correspondante de systèmes dynamiques hybrides (*relation* V.12).

Nous avons montré la liaison qui existe entre le comportement optimal et la stabilité d'une ligne TSS vue comme système dynamique hybride à mouvements discontinus, en précisant un ensemble invariant de cette dernière (*proposition P-V.1*).

Dans le cas des vitesses constantes de travail, en utilisant un critère de stabilité des systèmes dynamiques discrets, nous avons donné une autre démonstration – plus simple, lorsqu'elle se place cette fois-ci dans un contexte *déterministe* – de la condition suffisante – la condition du "bon ordre" des ouvriers sur la ligne – pour l'obtention du comportement optimal du point de vue de l'équilibrage (une nouvelle démonstration du *théorème T-V.3*). Dans ce cas on obtient le meilleur taux de production, qui peut être exactement calculé : il dépend des vitesses de travail des ouvriers.

La *poursuite* de ce travail vise à approfondir certaines directions qui n'ont été que suggérées.

Malgré leur simplicité, on estime que les deux algorithmes, $\mathtt{A}_1(\Omega)$ et $\mathtt{A}_2(\Omega)$, basés sur la vérification de la propriété Π d'un ensemble de gammes Ω, n'ont pas une complexité polynomiale. Ce qui conduit à la nécessité d'optimiser leur implémentation autant que possible.

La mise en évidence d'une forme générale d'écriture d'une gamme qui respecte un graphe de précédence peut constituer un résultat valorisant pour une nouvelle méthode de découpage en postes. Une telle méthode devrait s'appuyer sur la propriété des composantes connexes du graphe d'indifférence d'avoir la structure de poste de travail.

La résolution du problème d'équilibrage à l'aide de la programmation dynamique discrète n'a pas pris en compte l'existence potentielle des contraintes de précédence disjonctives (il s'agit dans ce cas d'un hypergraphe de précédence). Le traitement de ce type de contraintes a été mis en cause par quelques travaux récents, au sujet de la conception des systèmes d'assemblage multiproduits.

Remarquons que l'analyse des lignes d'assemblage avec auto-équilibrage par l'intermédiaire de la théorie des systèmes dynamiques hybrides a été possible sous quelques hypothèses assez restrictives, même si réalistes. Le calcul du temps de cycle et du taux de production est assez difficile, même dans les cas les plus particuliers. L'analyse de tels systèmes devient compliquée si on prend en compte les aspects subjectifs résultant de l'utilisation d'opérateurs humains. Il s'agit des cas où les vitesses de travail ne sont plus constantes et/ou les temps d'exécution des tâches ne sont plus déterministes. Très probablement, ces cas exigent une approche par la théorie des systèmes chaotiques.

Annexe A

Lignes d'assemblage avec auto-équilibrage : principaux résultats théoriques [BART 96a]

THÉORÈME T-V.1 : *existence du point fixe*

Pour toute ligne TSS **il existe *un point fixe***, c'est à dire il existe des positions des ouvriers, telles que, si les ouvriers commencent aux positions $\underline{x}^*$, alors ils réinitialisent toujours aux positions $\underline{x}^*$:

$$\underline{x}^* = f(\underline{x}^*)$$

Démonstration :

En principe, il s'agit de montrer ***la continuité*** de la fonction *f*.

Nous étendons *f* pour que son domaine de définition devienne l'entière "cellule" fermée de dimension *n*, $0 \le x_1 \le x_2 \le \ldots \le x_n \le 1$. Nous obtenons ainsi la fonction *g*, dont la continuité doit être démontrée.

Une fois que nous savons que *g* est continue, alors, conformément au *théorème de point fixe de Brouwer*, *g* admet un point fixe [BOLL 90], qui nécessairement appartient au domaine "naturel" de *f*, puisque, par la logique de TSS, aucun point du domaine étendu ne peut rester fixe sous *g*.

Notons ce point fixe par x_0. Ensuite, le raisonnement est simple :

$$\left.\begin{array}{l} g \equiv f\big|_{\text{le domaine de } f} \\ g(x_0) = x_0 \\ x_0 \text{ appartient au domaine de } f \end{array}\right\} \Rightarrow g(x_0) = f(x_0) = x_0 \Leftrightarrow x_0 \text{ est point fixe de } f.$$

Il nous reste à montrer que *g* ***est continue***.

Dans ce but-là, nous considérons *deux versions de ligne TSS* qui fonctionnent *en parallèle*. Donc, chaque ouvrier a *deux instances* (deux copies) : une instance sur la première ligne et l'autre sur la deuxième ligne. Dans la figure A-1 nous avons représenté deux itérations successives des deux lignes parallèles.

Nous considérons que le fonctionnement parallèle des deux lignes est gouverné par la règle TSS révisée. Par rapport à la règle TSS initiale, cette nouvelle règle contient une

restriction supplémentaire : aucun ouvrier ne peut pas avancer si son "jumeau" n'est pas au moins aussi avancé.

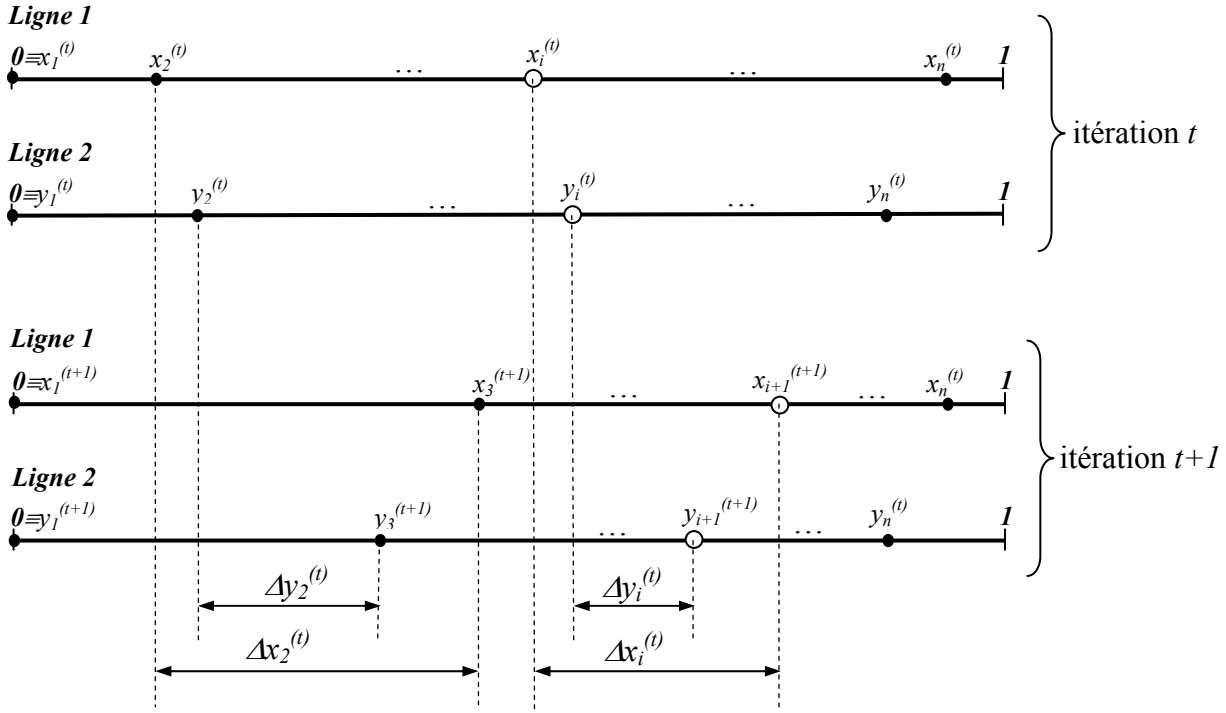


Fig. A-1. Deux lignes TSS fonctionnant en parallèle, *au début* de deux itérations successives

De cette restriction émerge la conclusion qu'un ouvrier avance dans une des deux situations :

- soit il essaie de rattraper son jumeau ;
- soit il avance en tandem avec son jumeau.

Dans la figure A-1 nous avons noté les positions des ouvriers sur la première ligne par $\{x_i^{(t)}\}_{i=1,2,\ldots n}$ et les positions sur la deuxième ligne par $\{y_i^{(t)}\}_{i=1,2,\ldots n}$. Les positions illustrées dans cette figure-là correspondent *au début* de deux itérations successives, *t* et *t+1*.

A l'égard de la même figure, nous avons utilisé également les **notations** suivantes :

$\Delta x_i^{(t)}$ = l'espace parcouru par l'ouvrier *i* sur la première ligne *au cours* de l'itération *t*

$\Delta y_i^{(t)}$ = l'espace parcouru par l'ouvrier *i* sur la deuxième ligne *au cours* de l'itération *t*

Nous allons démontrer par *récurrence en arrière* (pour *i=n,n-1,…1*) la proposition suivante :

P(*i*) : Sauf si la ligne réinitialise avant, les deux copies de l'ouvrier *i* arrivent au même poste au plus tard $T_i = \frac{1}{b} \cdot \sum_{j=i}^{n} | x_j^{(t)} - y_j^{(t)} |$ après le début de l'itération *t*, où $b = \inf_{i=\overline{1,n}, x \in [0,1]} \{v_i(x)\}$

Démonstration:

P(*n*): Sauf si la ligne réinitialise avant, les deux copies de l'ouvrier *n* arrivent au même poste au plus tard $T_n = \frac{1}{b} \cdot | x_n^{(t)} - y_n^{(t)} |$ après le début de l'itération *t*.

Notons par c_1 et c_2 les deux copies. Il peut exister deux cas :

- si les deux copies sont déjà au même poste, le temps qui nous intéresse est *0* ;
- sinon, supposons que la première copie travaille sur le *(k+1)*-ième poste et la deuxième copie travaille sur le *(l+1)*-ième poste :

$x_n^{(t)} \in [P_k, P_{k+1}]$, $y_n^{(t)} \in [P_l, P_{l+1}]$.

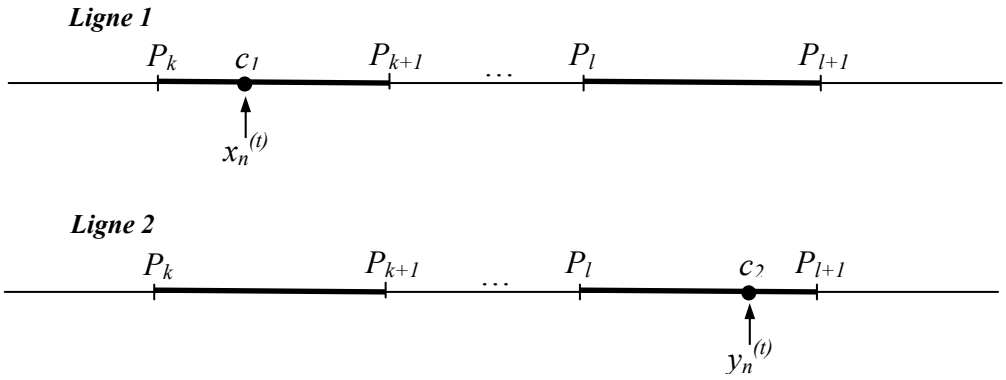


Fig. A-2. Les deux copies de l'ouvrier *n* travaillant aux postes différents : c_2 attend c_1

Sans perte de généralité, supposons, par exemple, que $l>k$ (voir la figure A-2). Alors c_2 attend pour que c_1 la rattrape. Donc, jusqu'à l'entrée sur le poste *(l+1)*, la variation de la position de c_1 sera

$P_l - x_n^{(t)} \le y_n^{(t)} - x_n^{(t)}$,

parcourue à une vitesse qui est à tout moment supérieure à b. Il résulte que c_1 va entrer au poste de c_2 au plus tard après $\frac{1}{b} \cdot | x_n^{(t)} - y_n^{(t)} |$. Par suite, **P**(*n*) est vraie.

Supposons que **P**(*k*) est vraie. Montrons que **P**(*k-1*) est aussi vraie.
Comme **P**(*k*) est vraie, il résulte qu'après au plus T_k les deux copies de l'ouvrier *k* avancent en tandem en travaillant au même poste. Par suite, la copie la moins avancée de l'ouvrier *k-1* – soit c_1 – ne sera pas bloquée en route vers le poste où se trouve la copie la plus avancée, c_2, qui l'attend. Conformément à un raisonnement analogue à celui de **P**(*n*), c_1 aura besoin du temps $\frac{1}{b} \cdot | x_{k-1}^{(t)} - y_{k-1}^{(t)} |$ au plus pour qu'elle arrive au poste de c_2.
Par conséquent, le temps total consommé pour que les deux copies de l'ouvrier *k-1* arrivent au même poste est au plus $T_k + \frac{1}{b} \cdot | x_{k-1}^{(t)} - y_{k-1}^{(t)} | = T_{k-1}$, ce qui signifie que **P**(*k-1*) est vraie.

Il résulte que **P**(*i*) est vraie pour $i = \overline{1,n}$.

Soit *i* de {*1,2,...n-1*} fixé.

Conformément à **P**(*i*), il est sûr qu'après au plus un temps T_i les deux copies de l'ouvrier *i*, c_1 et c_2, avancent en tandem en travaillant au même poste, soit le poste *r+1*. Dans la figure A-3 nous avons supposé l'existence d'un décalage initial entre les deux copies. Les notations $x_i^{(t)}$ et $y_i^{(t)}$ désignent leurs positions *au début* de l'itération *t*.

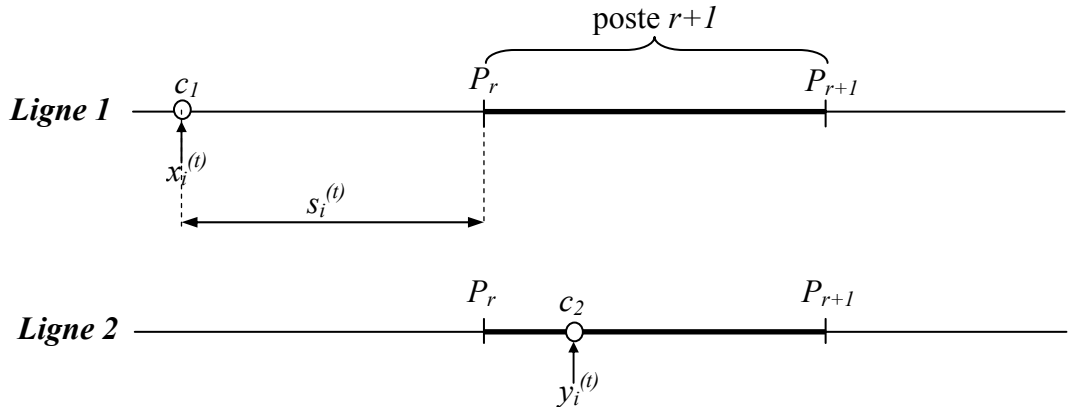


Fig. A-3. Les deux copies de l'ouvrier *i* – situation de décalage

Supposons que la deuxième copie, c_2, est plus avancée. Au début de l'itération t elle se trouve au poste $r+1$. Elle ne quitte pas cette position, $y_i^{(t)}$, jusqu'au moment où la première, c_1, va la rattraper.

La première copie va entrer dans le poste $r+1$ après avoir parcouru la distance $s_i^{(t)}$. Ce qui se passe au plus tard après T_i, conformément à **P**(i), si la ligne ne réinitialise pas avant. Ensuite, c_2 devra-t-elle encore attendre c_1, mais pas plus que le temps $\frac{1}{b} \cdot \left| y_i^{(t)} - s_i^{(t)} - x_i^{(t)} \right|$.

Le temps total, T_{t_i}, consommé pour que les deux copies atteignissent exactement la même position sur les deux lignes parallèles peut être ainsi estimé :

$$T_{t_i} \le T_i + \frac{1}{b} \cdot \left| y_i^{(t)} - s_i^{(t)} - x_i^{(t)} \right| \le T_i + \frac{1}{b} \cdot \left| y_i^{(t)} - x_i^{(t)} \right| \le 2 \cdot T_i$$

Mais il peut arriver que la ligne réinitialise avant que la copie la moins avancée rattrape la plus avancée ; ainsi, leurs positions de la fin de l'itération t peuvent éventuellement être différentes. Cette situation est illustrée dans la figure A-4 : pendant cette itération c_1 parcourt la distance totale $s_1^{(t)}$ et c_2 parcourt en total $s_2^{(t)}$.

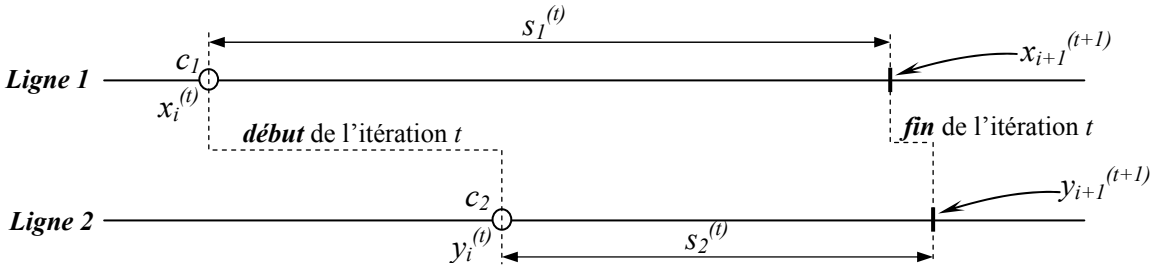


Fig. A-4. L'évolution des deux copies de l'ouvrier *i* pendant une itération : le décalage (éventuel) initial non récupéré au cours de l'itération

Rappelons que les positions *finales* de c_1 et c_2 coïncident respectivement avec les positions *initiales* des deux copies de l'ouvrier $i+1$ au commencement de l'itération suivante. Donc, nous pouvons écrire :

$$\left| x_{i+1}^{(t+1)} - y_{i+1}^{(t+1)} \right| = \left| x_i^{(t)} + s_1^{(t)} - y_i^{(t)} - s_2^{(t)} \right| = \left| \left(x_i^{(t)} - y_i^{(t)} \right) + \left(s_1^{(t)} - s_2^{(t)} \right) \right| \le$$

$$\le \left| x_i^{(t)} - y_i^{(t)} \right| + \left| s_1^{(t)} - s_2^{(t)} \right|$$

La distance $|s_1^{(t)}-s_2^{(t)}|$ peut être *au plus* la distance totale parcourue par la copie la moins lente, afin qu'elle rattrape sa "jumelle". Comme montré auparavant, cette distance nécessite *au plus* le temps $2 \cdot T_i$. Nous pouvons écrire :

$$\left| s_1^{(t)} - s_2^{(t)} \right| < 2 \cdot T_i \cdot B ,$$

où $B = \sup_{i=\overline{1,n},\, x\in[0,1]} \{v_i(x)\}$. Il s'ensuit que :

$$\text{(A.1)} \qquad \left| x_{i+1}^{(t+1)} - y_{i+1}^{(t+1)} \right| < 2 \cdot T_i \cdot B + \left| x_i^{(t)} - y_i^{(t)} \right|, \qquad \forall i = \overline{1, n-1}.$$

Revenons maintenant à l'expression de la fonction g :

$g(\underline{x}^{(t)})=\underline{x}^{(t+1)}$, où $\underline{x}^{(t)}=[x_1^{(t)}\ x_2^{(t)}\ \ldots\ x_n^{(t)}]$.

Tout en gardant les notations concernant les positions des ouvriers sur les deux lignes parallèles, nous voulons estimer la différence, en norme euclidienne, entre $g(\underline{x}^{(t)})$ et $g(\underline{y}^{(t)})$.

Nous savons que $x_1^{(t)}=y_1^{(t)}=0$. Alors, il est évident que :

$$\text{(A.2)} \qquad \left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\| = \left\| \underline{x}^{(t+1)} - \underline{y}^{(t+1)} \right\| \le \sum_{i=2}^{n} \left| x_i^{(t+1)} - y_i^{(t+1)} \right| .$$

Compte tenu de la relation (A.1), il résulte que :

$$\text{(A.3)} \qquad \sum_{i=2}^{n} \left| x_i^{(t+1)} - y_i^{(t+1)} \right| \le 2 \cdot B \cdot \sum_{i=1}^{n-1} T_i + \sum_{i=1}^{n-1} \left| x_i^{(t)} - y_i^{(t)} \right| .$$

Évidemment, $T_i<T_1$, $\forall\ i=2,3,\ldots n$. Par suite :

$$\sum_{i=1}^{n-1} T_i < \frac{n-1}{b} \cdot \sum_{i=1}^{n} \left| x_i^{(t)} - y_i^{(t)} \right| ,$$

relation que nous utilisons dans la relation (A.3), en obtenant que :

$$\sum_{i=2}^{n} \left| x_i^{(t+1)} - y_i^{(t+1)} \right| \le \frac{2 \cdot B}{b} \cdot (n-1) \cdot \sum_{i=1}^{n-1} \left| x_i^{(t)} - y_i^{(t)} \right| + \sum_{i=1}^{n-1} \left| x_i^{(t)} - y_i^{(t)} \right| .$$

Nous utilisons cette dernière relation dans la relation (A.2) :

$$\left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\| \le \sum_{i=2}^{n} \left| x_i^{(t+1)} - y_i^{(t+1)} \right| \le \left(\frac{2 \cdot B}{b} \cdot (n-1) + 1 \right) \cdot \sum_{i=1}^{n-1} \left| x_i^{(t)} - y_i^{(t)} \right| ,$$

qui conduit immédiatement à :

$$\text{(A.4)} \qquad \left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\| \le \left(\frac{2 \cdot B}{b} \cdot (n-1) + 1 \right) \cdot \sum_{i=1}^{n} \left| x_i^{(t)} - y_i^{(t)} \right| .$$

Rappelons que nous devons montrer *la continuité* de la fonction g. En utilisant la définition de la continuité, nous avons à montrer que :

$$\exists \delta = \delta(\varepsilon) : \left(\left(\forall \varepsilon > 0 : \left\| \underline{x}^{(t)} - \underline{y}^{(t)} \right\| < \varepsilon \right) \Rightarrow \left(\left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\| < \delta \right) \right).$$

Nous écrivons successivement :

$$\left.\begin{aligned} &\varepsilon > \left\| \underline{x}^{(t)} - \underline{y}^{(t)} \right\| \ge \max_{i=1,n} \left| x_i^{(t)} - y_i^{(t)} \right| \\ &\sum_{i=1}^{n} \left| x_i^{(t)} - y_i^{(t)} \right| < n \cdot \max_{i=1,n} \left| x_i^{(t)} - y_i^{(t)} \right| \end{aligned}\right\} \Rightarrow \sum_{i=1}^{n} \left| x_i^{(t)} - y_i^{(t)} \right| < n \cdot \varepsilon ,$$

résultat que nous utilisons dans la relation (A.4). Par conséquent :

$$\left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\| \le \left(\frac{2 \cdot B}{b} \cdot (n-1) + 1 \right) \cdot \sum_{i=1}^{n} \left| x_i^{(t)} - y_i^{(t)} \right| < \underbrace{\left(\frac{2 \cdot B}{b} \cdot (n-1) + 1 \right) \cdot n \cdot \varepsilon}_{\stackrel{\Delta}{=} \delta = \delta(\varepsilon)}$$

Donc, la norme $\left\| g(\underline{x}^{(t)}) - g(\underline{y}^{(t)}) \right\|$ devient petite chaque fois où la norme $\left\| \underline{x}^{(t)} - \underline{y}^{(t)} \right\|$ devient petite, ce qui signifie que la fonction *g* **est continue**. En revenant aux considérations du début de la démonstration, *g* possède **un point fixe**, qui est point fixe aussi pour la fonction *f*, qui décrit *l'orbite* (ou la trajectoire) d'une ligne TSS.

La démonstration est ainsi finie.

THÉORÈME T-V.2 : *convergence vers le point fixe unique*

> Pour toute ligne TSS, si les ouvriers sont rangés du plus lent au plus rapide, alors toute orbite (trajectoire) des positions des ouvriers ***converge*** vers l'unique point fixe.

Démonstration – grandes lignes :

1. Remarquer que chaque allocation successive d'un ouvrier *i* est une ***combinaison*** d'allocations de l'itération précédente.

2. Modéliser l'évolution des allocations par l'intermédiaire d'une ***chaîne de Markov finie***, avec probabilités de transition non stationnaires, mais ayant une structure spéciale.

3. Utiliser la chaîne de Markov pour montrer la ***convergence de la dernière allocation***, c'est à dire la convergence de la suite $\left\{ a_n^{(t)} \right\}_{t=0}^{\infty}$.

4. Montrer la ***convergence des positions de réinitialisation***, c'est à dire la convergence des suites $\left\{ x_i^{(t)} \right\}_{t=0}^{\infty}$, pour *i=1,2,...n*.

Résultats intermédiaires

Propriétés des allocations

(définies dans V.2.3.1)

a) l'allocation $a_i^{(t)}$ (de l'ouvrier *i* à l'itération *t*) est *continue* en $x_i^{(t)}$ et en $x_{i+1}^{(t)}$, excepté éventuellement en P_k, *k=1,2,...m-1* ;

b) si les ouvriers sont rangés du plus lent au plus rapide – $v_1 \prec v_2 \prec \ldots \prec v_n$ – alors $a_i^{(t)}$ est :

- *décroissante* en $x_i^{(t)}$
- et *strictement croissante* en $x_{i+1}^{(t)}$;

c) l'allocation du dernier ouvrier, $a_n^{(t)}$, est toujours *simple*, représentant la durée entre la sortie du *t*-ième article et celle du (*t+1*)-ième de la ligne ;

d) pour les allocations correspondant à un point fixe, il est vrai que $a_i^* \le a_n^*$, avec inégalité stricte seulement si l'ouvrier *i+1* est bloqué au commencement de l'itération.

Définition A-1 :

$$\rho = \max_{i=1,n-1}\left\{ \sup_{x\in[0;1]}\left\{ \frac{v_i(x)}{v_{i+1}(x)} \right\} \right\}$$

Lemme L-A.1 :

Si $x_n^{(t)} > x_n^{(t+1)}$, alors il existe $\lambda \in [0,\rho]$ tel que :

$$a_n^{(t+1)} = \lambda \cdot a_{n-1}^{(t)} + (1-\lambda)\cdot a_n^{(t)}$$

Introduisons l'allocation triviale : $a_0^{(t)}=0$, pour toute itération t.

Définissons une chaîne de Markov sur les états *0, 1, 2,…n*, où l'état *i* à l'itération *t* correspond à l'allocation de l'ouvrier *i* à une certaine itération (à expliquer plus bas). Nous appelons un état *i* à l'itération *t simple* ou *retardé*, si l'allocation correspondante, $a_i^{(t)}$, est *simple* ou *retardée*. Cette chaîne modélise comment les valeurs des allocations changent d'une itération à la suivante.

À cette chaîne nous associons *la matrice des probabilités de transition*, qui évolue en fonction des itérations. Notons par $T^{(t+1)}$ la matrice de dimension $(n+1)\cdot(n+1)$, dont les éléments $t_{ij}^{(t+1)}$ représentent les probabilités de transition de l'état *i* vers l'état *j* à l'itération *t+1*, *i=1,2,…n+1*, *j=1,2,…n+1*.

Soit *t* fixé.

Définition A-2 : ***matrice $T^{(t+1)}$ des probabilités de transition***

	0	…	**i-1**	**i**	…	**n-1**	**n**		
0	1	…	0	0	…	0	0		
.									
.									
.									
	$1-\frac{a_i^{(t+1)}}{a_n^{(t)}}$	…	0	0	…	0	$\frac{a_i^{(t+1)}}{a_n^{(t)}}$	*état simple*	$x_i^{(t)} \le x_i^{(t+1)}$
	$1-\frac{a_i^{(t+1)}}{a_{i-1}^{(t)}}$	…	$\frac{a_i^{(t+1)}}{a_{i-1}^{(t)}}$	0	…	0	0		$x_i^{(t)} > x_i^{(t+1)}$
i	$1-\frac{a_i^{(t+1)}}{a_n^{(t)}}$	…	0	0	…	0	$\frac{a_i^{(t+1)}}{a_n^{(t)}}$	*état retardé*	$(x_i^{(t)} \le P_{k-1}) \wedge (x_{i+1}^{(t)} \le x_{i+1}^{(t+1)})$
	$1-\frac{\alpha_i^{(t+1)}}{\rho}$	…	0	$\alpha_i^{(t+1)}$	…	0	$\frac{(1-\rho)}{\rho}\cdot\alpha_i^{(t+1)}$		$(x_i^{(t)} \le P_{k-1}) \wedge (x_{i+1}^{(t)} > x_{i+1}^{(t+1)})$
	$1-\frac{\beta_i^{(t+1)}}{\rho}$	…	$\beta_i^{(t+1)}$	0	…	0	$\frac{(1-\rho)}{\rho}\cdot\beta_i^{(t+1)}$		$(x_i^{(t)} > P_{k-1})$
.									
.									
.									
n	$1-\frac{a_n^{(t+1)}}{a_n^{(t)}}$	…	0	0	…	0	$\frac{a_n^{(t+1)}}{a_n^{(t)}}$		$x_n^{(t)} \le x_n^{(t+1)}$
	0	…	0	0	…	$\gamma_n^{(t+1)}$	$1-\gamma_n^{(t+1)}$		$x_n^{(t)} > x_n^{(t+1)}$

Nous avons utilisé les **notations** :

$$\left.\begin{aligned} \alpha_i^{(t+1)} &= \frac{a_i^{(t+1)}}{\rho \cdot a_i^{(t)} + (1-\rho)\cdot a_n^{(t)}} \\ \beta_i^{(t+1)} &= \frac{a_i^{(t+1)}}{\rho \cdot a_{i-1}^{(t)} + (1-\rho)\cdot a_n^{(t)}} \\ \gamma_n^{(t+1)} &= 1 - \frac{\tau_n(x_n^{(t+1)}, x_n^{(t)})}{\tau_{-1n}(x_n^{(t+1)}, x_n^{(t)})} \end{aligned}\right\} \quad i = \overline{1, n-1}.$$

Notons ensuite par $\underline{a}^{(t)} = [a_0^{(t)}\ a_1^{(t)}\ \ldots\ a_n^{(t)}]^T$ le vecteur des allocations des ouvriers à l'itération *t* (complété par l'allocation triviale, $a_0^{(t)}=0$).

Lemme L-A.2 :

La matrice $T^{(t+1)}$ des probabilités de transition est *bien définie*, *stochastique* et vérifie la relation :

(A.5) $$\underline{a}^{(t+1)} = T^{(t+1)} \cdot a^{(t)} = T^{(t+1)} \cdot T^{(t)} \cdot \ldots \cdot T^{(1)} \cdot \underline{a}^{(0)}.$$

Remarque :

La chaîne de Markov exécute les transitions ***en ordre inverse*** des itérations de la ligne TSS. Par exemple, la première transition est dictée par la matrice $T^{(t+1)}$, la seconde par $T^{(t)}$ et ainsi de suite.

Lemme L-A.3 : *convergence de la dernière allocation*

La suite $\left\{a_n^{(t)}\right\}_{t=0}^{\infty}$ converge vers une constante positive.

La démonstration utilise le raisonnement par l'absurde.

Remarque :

La convergence de l'allocation du dernier ouvrier implique *la convergence du temps de cycle* (du taux de production).

Lemme L-A.4 : *convergence des positions de réinitialisation*

$\lim_{t\to\infty} x_i^{(t)}$ existe pour chaque $i=1,2,\ldots n$.

La démonstration utilise la récurrence selon *i*.

THÉORÈME T-V.3 :

Si les vitesses sont *constantes* $v_i(x)=v_i$ et $v_1<v_2<\ldots.<v_n$ et si, de plus, les ouvriers ne sont *jamais bloqués*, alors la ligne converge *exponentiellement* vers *l'unique point fixe* pour lequel :

- l'ouvrier *i* répète l'exécution d'une portion de travail $\left[\frac{\sum_{j=1}^{i-1} v_j}{\sum_{j=1}^{n} v_j}, \frac{\sum_{j=1}^{i} v_j}{\sum_{j=1}^{n} v_j}\right]$;

- le taux de production est $\sum_{j=1}^{n} v_j$, le meilleur possible.

Démonstration :

Pratiquement, on se place dans un cas particulier du théorème T-V.2 : les ouvriers sont "bien rangés" et, de plus, il n'y a pas de blocages. C'est ainsi que la démonstration de ce théorème est en fait *une variante simplifiée* de celle du théorème antérieur.

Lorsqu'il n'existe ***pas de blocages***, les positions de réinitialisation changent d'une itération à la suivante selon les équations :

$$\text{(A.6)} \qquad \begin{cases} x_1^{(t+1)} = 0 \\ x_i^{(t+1)} = x_{i-1}^{(t)} + \dfrac{1 - x_n^{(t)}}{v_n} \cdot v_{i-1}, \quad i=1,2,...n, \end{cases}$$

qui décrivent ***un système dynamique linéaire***. De plus, toutes les allocations sont *simples* et peuvent être exprimées par une relation unitaire :

$$\text{(A.7)} \qquad a_i^{(t+1)} = \frac{x_{i+1}^{(t+1)} - x_i^{(t+1)}}{v_i}, \quad i = \overline{1,n-1},$$

que l'on utilise pour écrire la relation (A.6) en termes d'allocations $\underline{a}^{(t)}$:

$$\begin{cases} a_1^{(t+1)} = a_n^{(t)} \\ a_i^{(t+1)} = \dfrac{v_{i-1}}{v_i} \cdot a_{i-1}^{(t)} + (1 - \dfrac{v_{i-1}}{v_i}) \cdot a_n^{(t)}, \quad i=1,2,...n. \end{cases}$$

Une forme plus compacte de la relation ci-dessus fait intervenir la matrice **T**. De cette façon on arrive à une forme particulière de la relation (A.5) :

$$\underline{a}^{(t+1)} = \mathbf{T} \cdot \underline{a}^{(t)},$$

où nous avons gardé la notation du vecteur "étendu" des allocations. Remarquons que **T** est une particularisation de la matrice $T^{(t+1)}$ (voir définition A.2). Cette fois-ci, la matrice **T** décrit l'évolution d'une ***chaîne de Markov irréductible et apériodique*** (les probabilités de transition ne dépendent plus de temps, elles deviennent *stationnaires*).

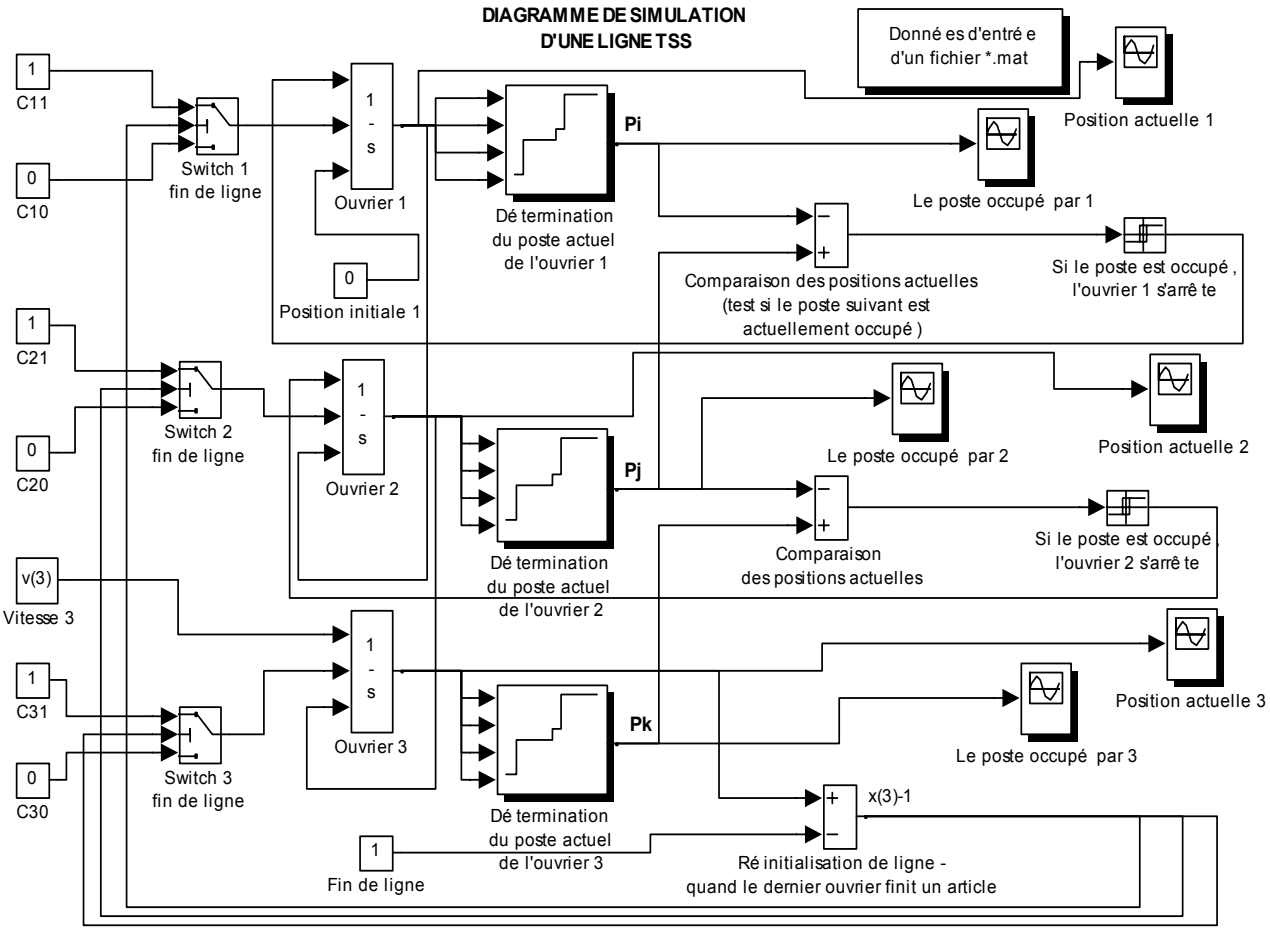
Par conséquent, tous les ouvriers arrivent à travailler le même intervalle de temps, qui est en fait *le temps de cycle* de la ligne TSS, noté par T_c :

$$\text{(A.8)} \qquad \lim_{t\to\infty} \underline{a}^{(t)} = \underbrace{[0 \quad T_c \quad T_c \quad ... \quad T_c]}_{n+1}^{T}, \qquad T_c = \frac{1}{\sum_{i=1}^{n} v_i}.$$

Des calculs simples montre que la relation (A.8), combinée avec (A.7), conduit à la première partie de la conclusion de ce théorème. La deuxième partie résulte de l'observation que le taux de production est égal à $\frac{1}{T_c}$.

Annexe B

Le schéma de simulation d'une ligne TSS, implémenté en Simulink



Annexe C

Systèmes dynamiques hybrides à mouvements discontinus [YE 95]

C.1 Notations et définitions

Définition C-1 : ***espace de temps***

Un espace métrique (T, ρ) s'appelle **espace de temps** si :

a) T est totalement ordonné (par la relation "$\prec$") ;

b) T contient un élément minimal $t_{min} \in T, \forall\, t \in T, t \neq t_{min} : t \prec t_{min}$;

c) $\forall\, t_1, t_2, t_3 \in T, t_1 \prec t_2 \prec t_3 :\ \rho(t_1, t_3) = \rho(t_1, t_2) + \rho(t_2, t_3)$;

d) T n'est pas supérieurement borné : $\forall\, M>0, \exists\, t \in T : \rho(t, t_{min})>M$.

Définition C-2 : ***mouvement***

Soit (X, d) un espace métrique et soit $A \subset X$.

Soit (T, ρ) un espace de temps et soit $T_0 \subset T$.

$\forall\, a \in A, \forall\, t_0 \in T_0$ on appelle **mouvement** une fonction

$$p(\cdot, a, t_0) : T_{a,t_0} \to X$$

si $p(t_0, a, t_0) = a$, où $T_{a,t_0} = \{t \in T : t_0 \preceq t, \rho(t, t_0) > l\}$ et $l>0$ peut être fini ou infini.

Observation :

X signifie l'espace d'états, dont A signifie l'ensemble d'*états initiaux*. T signifie l'espace de temps, dont T_0 signifie l'ensemble d'*instants initiaux*.

Définition C-3 : ***système dynamique hybride***

Soit S une famille de mouvements :

$$S \subset \{p(\cdot,a,t_0) \in \Lambda : p(t_0,a,t_0) = a\},$$

où $\Lambda = \bigcup_{(a,t_0)\in A\times T_0} \{T_{a,t_0} \times \{a\} \times \{t_0\} \to X\}$ est une réunion de fonctions définies sur $T_{a,t_0} \times \{a\} \times \{t_0\}$ à valeurs dans X.

Le 5-uple $\{T, X, A, S, T_0\}$ s'appelle **système dynamique hybride**.

Définition C-4 : ***invariant d'un système dynamique hybride***

Soit $\{T, X, A, S, T_0\}$ un système dynamique hybride.

Un ensemble $M \subset A$ s'appelle **invariant** du système S si

$$\forall t \in T_{a,t_0}, \forall t_0 \in T_0, \forall p(\cdot,a,t_0) \in S : a \in M \Rightarrow p(t,a,t_0) \in M .$$

On dit que M est un invariant de S ou (S, M) est invariant.

Définition C-5 : ***point d'équilibre***

$x_0 \in A$ s'appelle **point d'équilibre** d'un système dynamique hybride $\{T, X, A, S, T_0\}$ si l'ensemble $\{x_0\}$ est un invariant du système S.

Définition C-6 : ***stabilité au sens de Lyapounov***

Soit $\{T, X, A, S, T_0\}$ un système dynamique hybride et soit $M \subset A$ un invariant de S. Soit $x \in A$ fixé.

a) On dit que (S, M) est **stable** si

$$\forall \varepsilon > 0, \forall t_0 \in T_0 \ \ \exists \partial = \partial(\varepsilon, t_0) > 0 :$$
$$d(a,M) < \partial \Rightarrow d(p(t,a,t_0),M) < \varepsilon, \quad \forall t \in T_{a,t_0}, \forall p(\cdot,a,t_0) \in S$$

b) Si $\delta = \delta(\varepsilon)$, on dit que ($S$, M) est **uniformément stable**.

c) Si (S, M) est stable et si

$$\forall t_0 \in T_0 \ \exists \eta = \eta(t_0) :$$
$$d(a,M) < \eta \Rightarrow \lim_{t\to\infty} d(p(t,a,t_0),M) = 0, \forall p(\cdot,a,t_0) \in S$$

on dit que (S, M) est **asymptotiquement stable**.

d) Si (S, M) est uniformément stable et si

$$\exists \partial > 0 \text{ et } \forall \varepsilon > 0 \ \exists \tau = \tau(\varepsilon) : \quad d(a,M) < \partial \Rightarrow d(p(t,a,t_0),M) < \varepsilon,$$
$$\forall t \in \{t \in T_{a,t_0} : d(t,t_0) \geq \tau\}, \ \forall p(\cdot,a,t_0) \in S$$

on dit que (S, M) est **uniformément asymptotiquement stable**.

Observation :

Les définitions ci-dessus sont des adaptations naturelles des concepts correspondants rencontrés dans la théorie classique des systèmes dynamiques.

C.2 Stabilité des invariants

Nous considérons (*S*, *M*) invariant.

Idée de base :

1 – *englober* le système dynamique hybride $\{T, X, A, S, T_0\}$ – défini sur un espace de temps quelconque T – dans le système dynamique $\{\mathbf{R}^+, X, A, \overline{S}, R_0^+\}$ (défini sur $\mathbf{R}^+$) ;

2 – les propriétés de stabilité de (*S*, *M*) peuvent être déduites à partir des propriétés de stabilité de $(\overline{S}, M)$;

3 – établir des propriétés de stabilité pour le système $\{\mathbf{R}^+, X, A, \overline{S}, R_0^+\}$, (à mouvements discontinus).

Définition C-7 : ***englobement d'un espace de temps***

Soit (T, ρ) un espace de temps. Une fonction $g : T \to \mathbf{R}^+$ avec les propriétés :

a) $g(t_{min}) = 0,\ t_{min}\ est\ l'élément\ minimal\ de\ T$;

b) $g(t) = \rho(t, t_{min})\ pour\ t \neq t_{min}$

s'appelle **fonction d'englobement de l'espace de temps** T dans $\mathbf{R}^+$.

Observation :

Si nous notons $R_1 = g(T)$, alors g est une isométrie de T à R_1.

Définition C-8 : ***englobement d'un mouvement (par rapport à un état initial fixé)***

Soit $\{T, X, A, S, T_0\}$ un système dynamique hybride et soit $x \in A$ fixé.

Soit $g : T \to \mathbf{R}^+$ une fonction d'englobement au sens de la définition C-7 et $R_1 = g(T)$.

Soit $p(\cdot, a, t_0) \in S$ un mouvement défini sur T, c'est à dire $p(\cdot, a, t_0) : T \to X$.

Une fonction $\overline{p}(\cdot, a, r_0) : \mathbf{R}^+ \to X$ avec les propriétés suivantes :

a) $r_0 = g(t_0)$;

b) $\overline{p}(r, a, r_0) = \begin{cases} p(g^{-1}(r), a, t_0), & si\ r \in R_1 \\ x, & sinon \end{cases}$

s'appelle **fonction d'englobement du mouvement** $p(\cdot, a, t_0)$ de T à $\mathbf{R}^+$ **par rapport à l'état fixé** x.

Définition C-9 : ***englobement d'un système dynamique hybride***

Soit $\{T, X, A, S, T_0\}$ un système dynamique hybride défini sur un espace de temps abstrait T et soit $x \in A$.

Le système dynamique $\left\{\mathbf{R}^+, X, A, \overline{S}, R_0^+\right\}$ s'appelle **l'englobement de** $\{T, X, A, S, T_0\}$ **de** *T* **à** $\mathbf{R}^+$ **par rapport à** *x*, où :

$R_0^+ = g(T_0)$ et

$$\overline{S} \stackrel{\Delta}{=} \left\{\overline{p}(\cdot, a, t_0) : \overline{p}(\cdot, a, t_0) \textit{ est l'englobement de } p(\cdot, a, t_0) \textit{ par rapport} \text{ à } x,\ p(\cdot, a, t_0) \in S\right\}$$

Observations :

a) *Différents* choix de *x* conduisent à *différents englobements* d'un système donné S.

b) Soit *M* un invariant du système *S*. Soit $x_1 \in M, x_2 \in M$ et $\left\{\mathbf{R}^+, X, A, \overline{S}_1, R_0^+\right\}$, $\left\{\mathbf{R}^+, X, A, \overline{S}_2, R_0^+\right\}$ les englobements de *S* par rapport à x_1 et, respectivement, x_2. Alors $(\overline{S}_1, M)$ et $(\overline{S}_2, M)$ ont des propriétés de stabilité identiques.

Proposition :

Soit $\{T, X, A, S, T_0\}$ un système dynamique hybride et soit $M \subset A$ un invariant du système *S*. Soit *x* un point quelconque fixé de *M*.

Soit $\left\{\mathbf{R}^+, X, A, \overline{S}, R_0^+\right\}$ l'englobement de $\{T, X, A, S, T_0\}$ de *T* à $\mathbf{R}^+$ par rapport à *x*.

Alors *M* est aussi un invariant de $\overline{S}$ et, de plus, (S, M) et $(\overline{S}, M)$ possèdent des propriétés de stabilité identiques.

Annexe D

Critères de stabilité des systèmes dynamiques discrets [VOIC 86]

Définition D-1 : ***polynôme convergent***

Un polynôme de variable complexe z :

(D.1) $\Delta(z) = a_0 \cdot z^n + a_1 \cdot z^{n-1} + ... + a_1 \cdot z + a_0$,

à coefficients réels – $a_i \in \mathbf{R}$, $i=0,1,...n$, $a_0>0$ – est dit *convergent* si toutes ses racines sont strictement placées dans le cercle unité : $|z|<1$.

THÉORÈME T-D.1 : *condition* ***nécessaire et suffisante*** *de* ***stabilité asymptotique*** *d'un système dynamique linéaire, discret et invariant en temps* - SLDI

Soit le SLDI décrit par l'équation d'état :

(D.2) $\underline{x}(k+1) = A \cdot \underline{x}(k), \qquad k \in \mathbf{N}, \ \underline{x} \in \mathbf{R}^n, \ A \in \mathbf{R}^{n \times n}$.

Le système décrit par (D.2) est *asymptotiquement stable* si et seulement si le polynôme caractéristique de sa matrice d'état, A, est convergent.

THÉORÈME T-D.2 : *condition* ***nécessaire*** *pour la convergence d'un polynôme*

Pour qu'un polynôme sous la forme (D.1) soit convergent, il est *nécessaire* que les conditions suivantes soient simultanément remplies :

a) $\Delta(1)>0$;

b) $(-1)^n \cdot \Delta(-1)>0$;

c) $a_0>|a_n|$.

THÉORÈME T-D.3 : *condition* ***nécessaire et suffisante*** *pour la convergence d'un polynôme* (**JURY-BLANCHARD**)

Un polynôme sous la forme (D.1) est convergent si et seulement si les conditions ci-dessous sont simultanément remplies :

$$\begin{cases} a_0 > |a_n|, \\ b_0 > |b_{n-1}|, \\ c_0 > |c_{n-2}|, \\ \dots \\ f_0 > |f_1|, \end{cases}$$

où les coefficients qui interviennent sont donnés par le ***tableau Jury-Blanchard***.

Le tableau JURY-BLANCHARD

	z^n	z^{n-1}	z^{n-2}	…	z^{n-i}	…	z^2	z^1	z^0
$\Delta(z)$	a_0	a_1	a_2		a_{n-i}		a_{n-2}	a_{n-1}	a_n
$D(z)$	a_n	a_{n-1}	a_{n-2}		a_i		a_2	a_1	a_0
$\Delta_1(z)$	b_0	b_1	b_2		b_{n-i-1}		b_{n-2}	b_{n-1}	
$D_1(z)$	b_{n-1}	b_{n-2}	b_{n-3}		b_i		b_1	b_0	
$\Delta_2(z)$	c_0	c_1	c_2		c_{n-i-2}		c_{n-2}		
$D_2(z)$	c_{n-2}	c_{n-3}	c_{n-4}		c_i		c_0		
⋮									
$\Delta_{n-2}(z)$	e_0	e_1	e_2						
$D_{n-2}(z)$	e_2	e_1	e_0						
$\Delta_{n-1}(z)$	f_0	f_1							
$D_{n-1}(z)$	f_1	f_0							
$\Delta_n(z)$	g_0								
$D_n(z)$	g_0								

où les coefficients $\{a_i\}_{i=0,1,\dots n}$ sont donnés par la relation (D.1) et les autres coefficients sont calculés conformément aux relations suivantes :

$$b_j = a_0 \cdot a_j - a_n \cdot a_{n-j}, \quad j = \overline{0, n-1} \ ;$$

$$c_k = b_0 \cdot b_k - b_{n-1} \cdot b_{n-k-1}, \quad k = \overline{0, n-2} \ ;$$

…

$$f_l = e_0 \cdot e_l - e_2 \cdot e_{2-l}, \quad l = 0,1 \ ;$$

$$g_0 = f_0^2 - f_1^2 \ .$$

THÉORÈME T-D.4 : *condition **suffisante** pour la convergence d'un polynôme*

Pour qu'un polynôme sous la forme (D.1) soit convergent, il *suffit* que :

$|a_0|>|a_1|+|a_2|+\dots+|a_n|$.

THÉORÈME T-D.5 : *condition **suffisante** pour la convergence d'un polynôme* (**KAKEYA**)

Pour qu'un polynôme sous la forme (D.1) soit convergent, il *suffit* que :

$a_0>a_1>a_2>\dots>a_{n-1}>a_n>0$.

Démonstration :

Nous savons que les coefficients du polynôme $\Delta(z) = a_0 \cdot z^n + a_1 \cdot z^{n-1} + ... + a_1 \cdot z + a_0$ respectent la relation $a_0>a_1>a_2>\ldots>a_{n-1}>a_n>0$.

Nous devons montrer que ce polynôme est convergent. Dans ce but, nous utilisons la condition nécessaire et suffisante fournie par le critère Jury-Blanchard, ce qui implique le calcul des coefficients du tableau correspondant.

Le tableau Jury-Blanchard contient $2 \cdot n$ lignes groupées en paires : la première ligne d'une paire générique p, $p=1,2,...n$, contient $n-p+1$ coefficients, $\{x_r\}_{r=0,1,...n-p}$, en ordre croissant de leurs indices, tandis que la deuxième ligne contient les mêmes coefficients en ordre inverse. Nous allons montrer par récurrence selon p, $p=1,2,...n$, la proposition suivante :

P(*p***)** : $x_0>x_1>...>x_{n-p}>0$.

P(*1***)** : $b_0>b_1>...>b_{n-1}>0$.

$$\left.\begin{array}{l} b_i = a_0 \cdot a_i - a_n \cdot a_{n-i} > a_0 \cdot a_{i+1} - a_n \cdot a_{n-i-1} = b_{i+1}, \quad i = \overline{0, n-2} \\ b_{n-1} = a_0 \cdot a_{n-1} - a_n \cdot a_1 \\ \left.\begin{array}{l} a_0 > a_1 > 0 \\ a_{n-1} > a_n > 0 \end{array}\right\} \Rightarrow a_0 \cdot a_{n-1} - a_n \cdot a_1 > 0 \end{array}\right\} \Rightarrow b_{n-1} > 0$$

$$\Rightarrow b_0 > b_1 > ... > b_{n-1} > 0.$$

La proposition **P(***1***)** est vraie.

Supposons maintenant que **P(***p***)** est vraie et montrons que **P(***p+1***)** est vraie.

P(*p+1***)** : $y_0>y_1>...>y_{n-p-1}>0$.

$$\left.\begin{array}{l} y_i = x_0 \cdot x_i - x_{n-p} \cdot x_{n-p-i} \overset{\mathbf{P}(p)}{>} x_0 \cdot x_{i+1} - x_{n-p} \cdot x_{n-p-(i+1)} = y_{i+1}, \quad i = \overline{0, n-p-2} \\ y_{n-p-1} = x_0 \cdot x_{n-p-1} - x_{n-p} \cdot x_1 \\ \mathbf{P}(p) \Rightarrow \left.\begin{array}{l} x_0 > x_1 > 0 \\ x_{n-p-1} > x_{n-p} > 0 \end{array}\right\} \Rightarrow x_0 \cdot x_{n-p-1} - x_{n-p} \cdot x_1 > 0 \end{array}\right\} \Rightarrow y_{n-p-1} > 0$$

$$\Rightarrow y_0 > y_1 > ... > y_{n-p-1} > 0.$$

La conclusion est que **P(***p***)** est valable pour $p=1,2,...n$. Par suite, les conditions :

$$\begin{cases} a_0 > |a_n|, \\ b_0 > |b_{n-1}|, \\ c_0 > |c_{n-2}|, \\ ... \\ f_0 > |f_1|, \end{cases}$$

sont remplies, ce qui, conformément au critère Jury-Blanchard, est équivalent à la convergence du polynôme $\Delta(z)$.

Bibliographie

1 [ALLA 87] Alla, H. – "Réseaux de Petri colorés et réseaux de Petri continus : Application à l'étude des systèmes à événements discrets", Thèse d'État, Institut National Polytechnique de Grenoble – INPG, Grenoble, France, Juin 1987.

2 [ALMG 89] Almgren, R. – "On knowledge based programming systems for flexible automatic assembly", Institute of Technology, Department of Mechanical Engineering, S-581 83, Linköping, Sweden, 1989.

3 [ANDR 83] Andreasen, M.M., Kalher, S. and T. Lund – *Design for Assembly*, I.F.S. Publications/Springer-Verlag, 1983.

4 [ANDR 88] Andreasen, M.M. and T. Ahn – *Flexible assembly systems*, I.F.S. Publications/Springer-Verlag, 1988.

5 [ARCU 66] Arcus, A.L. – "Comsoal, a computer method of sequencing operation for assembly lines", *International Journal of Product Research*, Vol. 4, No . 4, 1966, pp. 259-277.

6 [BALD 91] Baldwin, D.F., Abell, T.E. and M.C.M. Lui – "An integrated computer aid for generating and evaluating assembly sequences for mechanical products", *IEEE Journal of Robotics and Automation*, Vol. 7, No. 1, February 1991.

7 [BAPT 91] Baptiste, P., Cho, C.H., Favrel, J. et M. Zouhri - "Une caractérisation analytique des ordonnancements admissibles sous contraintes hétérogènes en flow-shop", *Journal Européen des Systèmes Automatisés (APII–JESA)*, Vol. 25, 1991, pp. 87-102.

8 [BARA 91] Barakat, O. – "Contribution à la modélisation et à la simulation orientée objet des systèmes flexibles de fabrication", Thèse de doctorat de l'Université de Franche-Comté, Besançon, France, 1991.

9 [BARR 91a] Barrault, M. et C. Teyssier – "Étude et réalisation d'un logiciel d'aide à l'implantation des ateliers d'assemblage", Projet de fin d'études, École Nationale Supérieure de Mécanique et des Microtechniques, Besançon, France, 1991.

10 [BARR 91b] Barrault, M. – "Calcul du temps de cycle d'un poste à partir de son implantation", Mémoire de DEA, Université de Franche-Comté, Besançon, France, 1991.

11 [BART 95] Bartholdi, J.J., Eisenstein, D.D., Jacobs-Blecha, C. and H.D. Ratcliff – "Design of a bucket brigade production line", June 12, 1995. (*www.isye.gatech.edu/faculty/john_bartholdi/bucket-brigades*).

12 [BART 96a] Bartholdi, J.J.and D.D. Eisenstein – "A production line that balances itself", *Operations Research*, Vol. 44, No. 1, 1996, pp. 21-34.

13 [BART 96b] Bartholdi, J.J.and D.D. Eisenstein – "The agility of bucket brigade production lines", *Proceedings of Conference on Flexible and Intelligent Manufacturing*, January 22, 1996.

14 [BART 98] Bartholdi, J.J.and D.D. Eisenstein – "Bucket brigades – A self-organizing scheme for sharing work", February 26, 1998. (*www.isye.gatech.edu/faculty/john_bartholdi/bucket-brigades*).

15 [BART 99a] Bartholdi, J.J., Eisenstein, D.D. and R.D. Foley – "Performance of bucket brigades when work is stochastic", February 13, 1998, revised January 21, 1999 (*www.isye.gatech.edu/faculty/john_bartholdi/bucket-brigades*).

16 [BART 99b] Bartholdi, J.J., Bunimovich, L.A. and D.D. Eisenstein – "Dynamics of 2- and 3-worker bucket brigade production lines", *Operations Research*, Vol. 47, No. 3, 1999, pp. 488-491.

17 [BAYB 86] Baybars, I. – "On currently practised formulations of the assembly line balancing problem", *Journal of Operations Management*, 1985, pp. 449-453.

18 [BELE 85] Belea, C. – *Teoria Sistemelor*, Vol. II: *Sisteme neliniare*, Editura Didactică şi Pedagogică, Bucureşti, 1985.

19 [BERG 70] Berge, C. – *Graphes et hypergraphes*, Monographies universitaires de mathématiques, Dunod, 1970.

20 [BOLL 90] Bollabás, B. – *Linear Analysis*, Cambridge University Press, 1990.

21 [BONN 91] Bonneville, F., Baptiste, P. and H. Manier – "Representation of a set of process plans for the real time control of a flexible assembly system", *Proceedings of the International AMSE Conference "Signals and Systems"*, Warsaw, Poland, 1991.

22 [BONN 94] Bonneville, F. – "Élaboration d'une base de données d'équipements pour la conception des systèmes d'assemblage réactifs", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1994.

23 [BOOT 76] Booth, K.S. – "Testing for the consecutive ones property graphs and graph planarity using PQ-tree algorithms", *Journal of Computer and System Science*, Vol. 143, 1976, pp. 335-379.

24 [BOUJ 84] Bourjault, A. – "Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires", Thèse d'État, Université de Franche-Comté, Besançon, France, 1984.

25 [BOUJ 90] Bourjault, A. – *Méthodologie de l'assemblage*, Support de cours, Université de Franche-Comté, Besançon, France, 1990.

26 [BOUR 90] Bourrières, J.-P. – "Contribution à la modélisation intégrée des systèmes robotisés d'assemblage", Thèse d'État, Université de Franche-Comté, Besançon, France, 1990.

27 [BRAN 98] Branicky, M.S., Borkar, V.S. and S.K. Mitter – "A Unified Framework for Hybrid Control: Model and Optimal Control Theory", *IEEE Transactions on Automatic Control*, Vol. 43, No. 1, pp. 31-45.

28 [BRAT 99] Bratcu, A. and V. Mînzu – "A Simulation Study of Self-balancing Production Lines", *Proceedings of the 1999 IEEE International Conference on Intelligent Engineering Systems – INES '99*, Stará Lesná, Slovakia, November 1-3, 1999, pp. 165-170.

29 [CAMP 89] Campagne, J.-P. et G. Caplat – "Élaboration automatiques des gammes d'assemblage", *Journal Européen des Systèmes Automatisés (APII–JESA)*, Vol. 23, No. 1, 1989, pp. 53-68.

30 [CARL 88] Carlier, J. et P. Chrétienne – *Problèmes d'ordonnancement : modélisation / complexité / algorithmes*, Masson, Paris, 1988.

31 [CHAP 88] Chappe, D. et A. Bourjault – "Utilisation des réseaux de Petri temporisés pour la conception et l'évaluation des systèmes d'assemblage automatisés", *Proceedings of the 12th IMACS World Congress on Scientific Computation*, Paris, 1988.

32 [CHEN 94] Chen, K. and J.-M. Henrioud - "Formal Generation of Assembly Precedence Graphs", *Proceedings of the 10th ISPE/IFAC International Conference on CAD/CAM, Robotics and Factories of the Future*, August 21, Ottawa, Canada, 1994, pp. 725-731.

33 [CHEN 96] Chen, K. - "Contribution à une méthode systématique de détermination des graphes de précédence pour l'assemblage", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1996.

34 [CHOW 90] Chow, W.M. – *Assembly line design: methodology and applications*, Marcel Deker, Inc., New York – Basel, 1990.

35 [COHE 85] Cohen, G., Dubois, D., Quadrat, J.-P. and M. Viot – A Linear-System Theoretic View of Discrete-Event Processes and Its Use for Performance Evaluation in Manufacturing, *IEEE Transactions on Automatic Control*, Vol. AC-30, No. 3, 1985.

36 [COHE 90] Cohen, G. *et al.* – "Linear Systems in [max,+] algebra", *Proceedings of the 29th IEEE Conference on Decision and Control*, Honolulu, Hawaii, U.S.A., December, 1990.

37 [COHE 91] Cohen, G. *et al.* - "Second Order Theory of Min-linear Systems and its Application to Discrete Event Systems", *Proceedings the 30th IEEE Conference on Decision and Control*, Brighton, England, December 1990.

38 [DANL 99] Danloy, J., Petit, F., Leroy, A., De Lit, P. and B. Rekiek – "A pragmatic approach for precedence graphs generation", *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning – ISATP '99*, Porto, Portugal, July 1999, pp. 387-392.

39 [DARE 89] Dar-el, E.M. - "MALB – A heuristic technique for balancing large single-model assembly lines", *AIIE Transactions*, Vol. 5, No. 4, 1989, pp. 343-356.

40 [DAVI 89] David, R. et H. Alla – *Du grafcet aux réseaux de Petri*, Hermès, Paris, 1989.

41 [DELC 89a] Delchambre, A., Coupez, D. and P. Gaspart – "Knowledge-based assembly by disassembling planning", *Proceedings of the International Conference on Expert Systems in Engineering Applications – ICESEA '89*, Wuhan, China, October 1989.

42 [DELC 89b] Delchambre, A. – "Data Structures for Computer-Aided Assembly Planning: A Survey", *INCOM 1989*, Madrid, Spain.

43 [DELC 90a] Delchambre, A. – "Conception assistée par ordinateur de gammes opératoires d'assemblage", Mémoire présenté en vue de l'obtention du grade de docteur en sciences appliquées, Université Libre de Bruxelles, Belgique, 1990.

44 [DELC 90b] Delchambre, A. – "A pragmatic approach to computer-aided assembly planning", *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, U.S.A., IEEE Computer Society Press, pp. 1600-1605.

45 [DELC 92] Delchambre, A. (editor) – *Computer-aided Assembly Planning*, Chapman & Hall, London, U.K., First Edition, 1992.

46 [DELC 96] Delchambre, A. (editor) – *CAD Method for Industrial Assembly: Concurrent Design of Products, Equipment and Control Systems*, John Wiley and Sons, Inc., Chichester, U.K., 1996.

47 [DINI 92] Dini, G. and M. Santochi – "Automated sequencing and subassembly detection in assembly planning", *Annals of the CIRP*, 41/1/1992:1-4.

48 [DUFR 91] Dufrène, L. – "Contribution à une méthodologie de conception des systèmes d'assemblage pour familles de produits", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1991.

49 [DUMI 97] Dumitrescu, D. şi H. Costin – *Reţele neuronale – teorie şi aplicaţii*, Editura Teora, Bucureşti, 1997.

50 [ELMA 89] Elmaraghy, H.A. and L. Knoll – "Design and automatic assembly sequence generation of a DC motor", *Proceedings of the 15th IAVD Conference*, Geneva, Switzerland, 1989.

51 [ENME 89] Enmer, A. - "Équilibrage des chaînes d'assemblage : bibliographie et proposition d'une méthode permettant le parallélisme", Mémoire de DEA, INSA, Lyon, France, 1989.

52 [FALK 93] Falkenauer, E. and A. Delchambre – "Ressource planning in the SCOPES Project", *Proceedings of the 26th International Symposium on Automotive Technology and Automation*, September 1993, Automotive Automation Limited, pp.295-302.

53 [FALK 96] Falkenauer, E. – "A Hybrid Grouping Genetic Algorithm for Bin Packing", *Journal of Heuristics*, Vol. 2, No. 1, 1996, pp. 5-30.

54 [FALK 98a] Falkenauer, E. – *Genetic Algorithms and Grouping Problems*, John Wiley and Sons, Inc., Chichester, U.K., First Edition, 1998.

55 [FALK 98b] Falkenauer, E. – "On method overfitting", *Journal of Heuristics*, Vol. 4, 1997, pp. 281-287.

56 [FARR 87] Farreny, H. et M. Ghallab – *Techniques de l'intelligence artificielle*, Hermès, Paris, 1987.

57 [FAZI 87] De Fazio, T.L. and D.E. Whitney – "Simplified generation of all mechanical assembly sequences", *IEEE Journal of Robotics and Automation*, Vol. 3, No. 6, December 1987.

58 [FAZI 88] De Fazio, T.L. and D.E. Whitney – Correction to "Simplified generation of all mechanical assembly sequences", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 6, December 1988.

59 [FAZI 99] De Fazio, T.L., Rhee, S.J. and D.E. Whitney – "Design-specific approach to design for assembly (DFA) for complex mechanical assemblies", *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 5, 1999, pp. 869-881.

60 [FIGO 88] Figour, J. - "Vue d'ensemble de l'assemblage automatisé", *Journées de microtechnique*, Lausanne, Suisse, 1988, pp. 39-55.

61 [FLAU 98] Flaus, J.-M. – "Modeling and analysis of hybrid dynamical systems: An introduction", *European Journal of Automation Systems*, Vol. 32, No. 7-8, 1998, pp. 797-830.

62 [FLEJ 93] Fleury, J.-P. – "Détection des sous-ensembles durant la détermination des gamme d'assemblage", Rapport de recherche DMT-IMT, Institut de Microtechnique, École Polytechnique Fédérale de Lausanne - EPF, Suisse, 1993.

63 [FLEG 95] Fleury, G. – "Applications des méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement", *Automatique, Productique, Informatique Industrielle – APII*, Vol. 29, No. 4-5, 1995, pp. 445-470.

64 [FOUD 99] Fouda Bana, P. – "Génération de gammes d'assemblage pour les familles de produits", Travail de spécialisation présenté dans le cadre du DES en productique, Université Libre de Bruxelles , Belgique, 1999.

65 [FREE 67] Freeman, J.R. and J.V. Jucker – "The line balancing problem", *Journal of Industrial Engineering*, Vol. 18.361, 1987.

66 [FREE 68] Freeman, J.R. –"A general line balancing model", *Proceedings of the 19th Annual Conference AIIE*, 1968, pp. 230-235.

67 [FROM 88] Frommherz, B. and J. Hornberger – "Automatic generation of precedence graphs", *Proceedings of the International Symposium on Industrial Robots*, April 1988, IFS Ltd. and authors, pp. 453-466.

68 [GENT 88a] Gentina, J.-C., Bourey, J.-P. and M. Kapusta - "Coloured adaptive structured Petri net: I. A tool for the automatic synthesis of hierarchical control of flexible manufacturing systems", *Systems Revue*, Vol. 1, No. 1, 1988, pp. 39-47.

69 [GENT 88b] Gentina, J.-C., Bourey, J.-P. and M. Kapusta - " Coloured adaptive structured Petri net: II. Deduction of the structured graph from the pregraph and application - modelling of a flexible workshop", *Systems Revue*, Vol. 1, No. 2, 1988, pp. 103-109.

70 [GHOS 89] Ghosh, S. and R. Gagnon - "A comprehensive literature review and analysis of the design, balancing and schedulingof assembly systems", *International Journal of Production Research*, Vol. 27, No. 4, 1989, pp. 637-670.

71 [GRAV 79] Graves, S.C. and D.E. Whitney - "A mathematical procedure for equipment selection and system evaluation in programmable assembly", *Proceedings of the 15th IEEE Conference on Decision and Control*, 1979, pp. 531-536.

72 [GRAV 83] Graves, S.C. and B.W. Lamar - "An integer programming procedure for assembly system design problems", *Operations Research*, Vol. 31, No. 3, 1983, pp. 522-545.

73 [GRAV 88] Graves, S.C. and C.H. Redfield - "Equipment selection and task assignment for multiproduct assembly system design", *International Journal of Flexible Manufacturing Systems*, 1988, pp. 31-50.

74 [GREW 90] Grewal, S., Kaebernick, H., Tran, P. and A. Choi - "Planning for assembly", *Proceedings of the 11th Congress on Assembly Automation*, Dearborn, Michigan, U.S.A.

75 [GUMO 80] Gumowski, I. et C. Mira – *Dynamique chaotique – transformations ponctuelles, transition, ordre-désordre*, Cepadues Editions, Collection Nabla, Toulouse, 1980.

76 [GUNT 83] Gunther, R.E., Johnson, G.D. and R.S. Peterson - "Currently practised formulations for the assembly line balancing problem", *Journal of Operation Management*, Vol. 3, No. 4, 1983, pp. 209-221.

77 [GUST 88] Gustavson, R.E. - "Design of cost-effective assembly systems. Succesful planning and implementation of flexible assembly systems", Ann Arbor, Michigan, U.S.A., March 29-31, 1988.

78 [GUST 90] Gustavson, R.E. - "S.P.M.: A connection from product to assembly system", The Charles Stark Draper Laboratory - CSDL, 555 Technology Square, Cambridge, Massachusetts, U.S.A., 1990.

79 [GUTJ 64] Gutjahr, A.L. and G.L. Nemhauser - "An Algorithm for the Line Balancing Problem", *Management Science*, Vol. 11, No. 2, 1964.

80 [HELD 89] Held, M., Karp, R.M. and R. Shareshian - "Assembly line balancing – dynamic programming with precedence constraints", *Operations Research*, 1962, pp. 442-459.

81 [HELG 61] Helgeson, W.B. and D.P. Birnie - "Assembly line balancing using the ranked positional weight technique", *Journal of Industrial Engineering*, Vol. XII, No. 6, 1961, pp. 394-398.

82 [HENR 89] Henrioud, J.-M. – "Contribution à la conceptualisation de l'assemblage automatisé", Thèse d'État, Université de Franche-Comté, Besançon, France, 1989.

83 [HENR 90] Henrioud, J.-M., Bonneville, F. and A. Bourjault – "Evaluation and Selection of Assembly Plans", *APMS '90*, Espoo, Finland, August 1990, pp. 212-219.

84 [HENR 91] Henrioud, J.-M. and A. Bourjault – "LEGA: A computer-aided generator of assembly plans". In: Homem de Mello and S. Lee (editors) – *Computer-Aided Mechanical Assembly Planning*, Chapter 8, Kluwer Academic Publishers, Norwell, Massachusetts, U.S.A., 1991.

85 [HENR 92] Henrioud, J.-M. and A. Bourjault – "Computer-aided assembly process planning", *Journal of Engineering Manufacture*, Vol. 206, Part B1, 1992, pp. 61-66.

86 [HENR 93] Henrioud, J.-M., Bourjault, A. and V. Mînzu – "Assembly process planning and assembly systems design", *Proceedings of the 3rd International Conference on Flexible Automation and Integrated Manufacturing – FAIM '93*, Limerick, Ireland, June 28-30, pp. 591-601.

87 [HENR 99] Henrioud, J.-M. and A. Bratcu – "Algorithm for generating the precedence graphs in assembly systems". In: N.E. Mastorakis (editor) – *Software and Hardware Engineering for the 21st Century*, World Scientific Engineering Society Press, Danvers, Massachusetts, U.S.A., 1999, pp. 44-49.

88 [HOLM 87] Holmes, C.A. - "Equipment selection and task assignment in an assembly system design problem", MSc. Dissertation, Operations Research Center ; Massachusetts Institute of Technology – MIT, U.S.A., 1987.

89 [HOME 88] Homem de Mello, L.S. and A.C. Sanderson – "Planning repair sequences using AND/OR representation of assembly plans", *IEEE International Conference on Robotics and Automation*, 1988, pp. 1861-1866.

90 [HOME 89] Homem de Mello, L.S. and A.C. Sanderson – "A correct and complete algorithm for the generation of mecahnical assembly sequences", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, pp. 56-61.

91 [HOME 90] Homem de Mello, L.S. and A.C. Sanderson - "AND/OR graph representation of assembly plans", *IEEE transactions on Robotics and Automation*, Vol. 6, No. 2, April 1990, pp. 188-199.

92 [HOME 91] Homem de Mello, L.S., Lee, S.- "Representations of mechanical assembly sequences", *IEEE transactions on Robotics and Automation*, Vol. 7, No. 2, April 1991, pp. 211-227.

93 [HONG 97] Hong, D.S. and H.S. Cho – "A genetic algorithm based approach to the generation of robotic assembly sequences", *IFAC Control Engineering Practice*, Vol. 7, No. 2, 1999, pp. 151-159.

94 [IGNA 65] Ignall, E.J. - "A review of assembly line balancing", *Journal of Industrial Engineering*, Vol. XVI, No. 4, 1965, pp. 244-254.

95 [JEAN 86] Jeannes, R. - "Méthodologie d'analyse des produits pour leur reconception en vue du montage automatisé", Thèse de doctorat, École Nationale des Arts et Métiers, 1986.

96 [JENT 83] Jentsch, W. and F. Kaden – "Automatic generation of assembly sequences", *Proceedings of the 3rd Conference on A.I. and Information Control Systems of Robots*, Amsterdam, The Netherlands, 1983.

97 [JUNG 97] Jung, L. - "A Single-Run Optimization Algorithm for Stochastic Assembly Line Balancing Problems", *Journal of Manufacturing Systems*, Vol. 16, No. 3, 1997, pp. 204-210.

98 [KILB 61] Kilbridge, M.D. - "A heuristic method of assembly line balancing", *Journal of Industrial Engineering*, Vol. XII, No. 4, 1961, pp. 292-299.

99 [KO 87] Ko, H.D. and K.W. Lee - "Automatic assembling procedure generation from mating conditions", *Computer-aided design*, Vol. 19, 1987, pp. 3-10.

100 [KUMA 92] Kumar, V. - "Algorithms for constraint-satisafaction problems: A survey", *AI Magazine*, Vol. 13, No. 1, 1992, pp. 32-44.

101 [KUSI 90] Kusiak, A. – *Intelligent Manufacturing Systems*, Prentice-Hall, 1990.

102 [KUSI 94] Kusiak, A. and R. Dorf (editors) – *Concurrent Engineering: Issues, Models and Solutions Approaches*, Chapter 3, John Wiley and Sons, Inc., New York, 1994, pp. 35-49.

103 [L.A.G. 90] "Modélisation et évaluation des performances des systèmes de production", *École d'été d'automatique de Grenoble*, Laboratoire d'Automatique de Grenoble, 1990.

104 [L.A.G. 91] "Modélisation, simulation, conduite des systèmes de production", *École d'été d'automatique de Grenoble*, Laboratoire d'Automatique de Grenoble, 1991.

105 [LEE 88] Lee, S.H. and Y.G. Shin - "Automatic construction of assembly partial-order graphs", *Proceedings of the 1988 International Conference on CIM*, 1988, pp. 383-392.

106 [LEVI 88] Levi, P. - "TOPAS: A task-oriented planner for optimized assembly sequences", *Proceedings of the 8th European Conference on Artificial Intelligence*, Munich, West Germany, 1988.

107 [LIT 00] De Lit, P. – "A comprehensive and integrated approach for the design of a product familly and its assembly system", Mémoire présenté en vue de l'obtention du grade de docteur en sciences appliquées, Université Libre de Bruxelles, Belgique, 2000.

108 [LUI 88] Lui, M.C.M. - "Generation and evaluation of mechanical assembly sequences using the liaison-sequence method", MSc. Dissertation, Mechanical Engineering Department, Massachusetts Institute of Technology – MIT, U.S.A., 1988.

109 [MACA 72] Macaskill, J.L.C. – "Production line balances for mixed-model lines", *Management Science*, Vol. 19, No. 4, Part I, December 1972, pp. 423-434.

110 [MASC 90] Mascle, C. - "Approche méthodologique de détermination de gammes par désassemblage", Thèse de doctorat, École Polytechnique Fédérale de Lausanne – EPF, Suisse, 1990.

111 [MAST 70] Mastor, A.A. - "An experimental investigation and comparative evolution of production line balancing technics", *Management Science*, Vol. 16, No. 11, 1970, pp. 728-746.

112 [MÎNZ 93] Mînzu, V. and J.-M. Henrioud – "Systematic method for the design of flexible assembly systems", *Proceedings of the International Conference on Robotics,* 1993, pp. 56-62.

113 [MÎNZ 95] Mînzu, V. – "Contribution à une approche systématique de découpage en postes des systèmes d'assemblage", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1995.

114 [MÎNZ 97] Mînzu, V. et J.-M. Henrioud – "Approche systématique de structuration en postes des systèmes d'assemblage monoproduits", *Journal Européen des Systèmes Automatisés (RAIRO-APII-JESA)*, Vol. 31, No. 1, 1997, pp.57-78.

115 [MÎNZ 98] Mînzu, V.and J.-M. Henrioud – "Stochastic algorithm for tasks assignment in single or mixed-model assembly lines", *Journal Européen des Systèmes Automatisés (APII-JESA)*, Vol. 32, No. 7-8, 1998, pp. 831-851.

116 [MÎNZ 99] Mînzu, V., A. Bratcu and J.-M. Henrioud – "Construction of precedence graphs equivalent to a given set of assembly sequences", *Proceedings of the 1999 International Symposium on Assembly and Task Planning – ISATP '99*, Porto, Portugal, July 1999, IEEE Computer Society Press, pp. 14-19.

117 [MOOD 65] Moodie, C.L. and H.H. Young - "A heuristic method of assembly line balancing for assumptions of constant or variable work element times", *Journal of Industrial Engineering*, Vol. XVI, No. 1, 1965, pp. 23-29.

118 [NAPH 99a] Naphade, K.S., Storer, R.H. and S.D. Wu - "Graph teoretic generation of assembly plans, Part i: Correct generation of precedence graphs", IMSE Technical Report 99T-003, Lehigh University, Bethlehem, Pennsylvania, U.S.A., 1999.

119 [NAPH 99b] Naphade, K.S., Wu, S.D. and R.H. Storer - "Graph teoretic generation of assembly plans, Part ii: Problem decomposition and optimization algorithms" IMSE Technical Report 99T-004, Lehigh University, Bethlehem, Pennsylvania, U.S.A., 1999.

120 [OLIV 86] Olivier, P. – "Analyse et synthèse des systèmes d'assemblage automatisés", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1986.

121 [PERR 92] Perrard, C. – "Contribution à une méthodologie d'aide à l'implantation et à la spécification des équipements des systèmes flexibles d'assemblage", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1992.

122 [PERR 93] Perrard, C., Lutz, P., Henrioud, J.-M. and A. Bourjault – "The MARSYAS software: An efficient tool for the rational design of assembly systems", *Proceedings of the International Conference on Assembly*, Adelaide, Australia, 1993, pp. 77-85.

123 [PINT 81] Pinto, P.A., Dannenbring, D.G. and B.M. Khumawala – "Branch-and-bound and heuristic procedures for assembly line balancing with paralleling of stations", *International Journal of Production Research*, Vol. 19, No. 5, 1981, pp. 565-576.

124 [PINT 83] Pinto, P.A., Dannenbring, G. and B.M. Khumawala - "Assembly line balancing with processing alternatives: An application", *Management Science*, Vol. 29, No. 7, July 1983.

125 [PREN 64] Prenting, T.O. and R.M. Battaglin – "The precedence diagram: A tool for analysis in assembly line balancing", *Journal of Industrial Engineering*, Vol. XV, No. 4, 1964, pp. 208-213.

126 [REKI 97] Rekiek, B., Falkenauer, E. and A. Delchambre – "Multi-Product Resource Planning", *Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning – ISATP '97*, Marina del Rey, California, U.S.A., August 7-9, 1997, pp. 115-121.

127 [REKI 99a] Rekiek, B., Pellichero, F., De Lit, P.Falkenauer, E. and A. Delchambre – "Towards physical layout of aseembly lines". In: N.E. Mastorakis (editor) – *Progress in Simulation, Modelling, Analysis and Synthesis of Modern Electrical and Electronic Devices and Systems*, World Scientific Engineering Society Press, Danvers, Massachusetts, U.S.A., 1999, pp. 307-312.

128 [REKI 99b] Rekiek B., De Lit, P., Pellichero, F., Falkenauer, E. and A. Delchambre – "Applying the equal piles problem to balance assembly lines", *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning – ISATP '99*, Porto, Portugal, July 1999, pp. 399-404.

129 [REMM 92] Remm, P. – "Contribution à l'évaluation et à la sélection des gammes d'assemblage", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1992.

130 [REND 95] Renders, J.-M. – *Algorithmes génétiques et réseaux de neurones – applications à la commande des processus*, Hermès, Paris, 1995.

131 [SCHA 78] Schrage, L. - "Dynamic programming solution of sequencing problems with precedence constraints", *Operations Research*, Vol. 26, No. 3, 1978, pp. 444-449.

132 [SCHO 91] Schroer, B.J., Wang, J. and M.C. Ziemke – "A look at TSS through simulation", *Bobbin magazine*, July 1991, pp. 114-119.

133 [SIMS 88] Simsik, D., Kovak, J. and L. Madarasz – "The application of expert systems in the design of robotic assembly lines", *Factory 2000: Integrating Information and Material Flow*, Cambridge, U.K., August 31 – September 1, 1988, pp. 213-221.

134 [SURE 96] Suresh, G., Vinod, V.V. and S. Sahu – "A genetic algorithm for assembly line balancing", *Production Planning & Control*, Vol. 7, No. 1, 1996, pp. 38-46.

135 [THOM 67] Thomopoulos, N.T. – "Line balancing sequencing for mixed-model assembly", *Management Science*, Vol.14, No. 2, 1967, pp. 59-75.

136 [THOM 70] Thomopoulos, N.T. – "Mixed-model line balancing with smothed station assignments", *Management Science*, Vol. 18, No. 9, May, 1970, pp. 593-603.

137 [TONG 61] Tonge, F.M. - "A heuristic program for assembly line balancing", *The Ford Foundation Doctoral Dissertation Series*, Prentice-Hall, 1961.

138 [VALL 87] Vallet, G. – "Équilibrage des lignes d'assemblage : étude bibliographique et présentation d'une méthode basée sur l'étude des chronogrammes", Mémoire de DEA, Université de Franche-Comté, Besançon, France, 1987.

139 [VALL 90] Vallet, G. – "Contribution au pilotage réactif d'installations flexibles d'assemblage automatique", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1990.

140 [VANA 78] Van Assche, F. and W.S. Herroelen – "An optimal procedure for the single-model deterministic assembly line balancing problem", *European Journal of Operations Research*, 3/1978, pp. 142-149.

141 [VOIC 86] Voicu, M. – *Tehnici de analiză a stabilității sistemelor automate*, Editura Tehnică, Bucureşti, 1986.

142 [WEUL 89] Weule, H. and T. Friedman – "Computer-aided product analysis in assembly planning", *Annals of the CIRP*, 38/1/1989:1-4.

143 [WHIT 86] Whitney, D.E. *et al.* – "Computer-aided design of flexible assembly systems", The Charles Stark Draper Laboratory First Report CSDL-R-1947, Cambridge, Massachusetts, U.S.A., 1986.

144 [WHIT 88] Whitney, D.E., De Fazio, T., Gustavson, R.E., Graves, S.C., Cooprider, K., Klein, C.J., Lui, M. and S. Pappu - "Computer-aided design of flexible assembly systems", The Charles Stark Draper Laboratory Final Report CSDL-R-2033, Cambridge, Massachusetts, U.S.A., 1988.

145 [WINT 85] De Winter, D. and H. Van Brussel - "An expert system for flexible assembly system design", *Robotic Trends*, 1985.

146 [WOLT 89] Wolter, J.D. – "On the automatic generation of assembly plans", *Proceedings of the 1989 International Conference on Robotics and Automation*, 1989, pp. 62-68.

147 [WU 87] Wu, L.C. and H.K. Edwards - "The design of single-model assembly lines", *IASTED '87*, New Orleans, U.S.A., 1987.

148 [YE 98] Ye, H., Michel, A.N. and L.Hou – "Stability Theory for Hybrid Dynamical Systems", *Proceedings of the 34th IEEE Conference on Decision and Control*, New Orleans, U.S.A., pp. 2679-2684.

149 [YIN 95] Yin, B. – "Contribution au pilotage réactif des systèmes flexibles d'assemblage : méthode d'équilibrage dynamique", Thèse de doctorat, Université de Franche-Comté, Besançon, France, 1995.

150 [YOSH 83] Yoshida, T., Mori, H. and H. Shigematsu – "Analytic study of chaos of the tent map: band structures, power spectra and critical behaviours", *Journal of Statistical Physics*, Vol. 31, No. 2, 1983, pp. 279-308.

Répertoire des figures

Chapitre I PROBLEMATIQUE DE L'ASSEMBLAGE

Chapitre II PROPRIETES DES GRAPHES DE PRECEDENCE

Chapitre III DETERMINATION DES GRAPHES DE PRECEDENCE A PARTIR D'UN ENSEMBLE DE GAMMES D'ASSEMBLAGE

Chapitre IV METHODES DE CONCEPTION DES SYSTEMES D'ASSEMBLAGE – AFFECTATION DES TACHES ET EQUILIBRAGE

Chapitre V SYSTEMES D'ASSEMBLAGE AVEC AUTO-EQUILIBRAGE

Annexe A *Lignes d'assemblage avec auto-équilibrage : principaux résultats théoriques* [BART 96a]

Annexe B *Le schéma de simulation d'une ligne TSS, implémenté en Simulink*

Index alphabétique

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

RÉSUMÉ

Cette thèse est une contribution à une démarche globale de conception rationnelle des systèmes d'assemblage. Elle concerne plus précisément le problème de génération des graphes de précédence, en vue de leur utilisation par les méthodes d'équilibrage des systèmes d'assemblage.

Le premier chapitre de ce travail est consacré à la description de la problématique des systèmes d'assemblage.

Le deuxième chapitre présente un état de l'art des approches de génération des graphes de précédence pour l'assemblage. Les propriétés de ceux-ci sont listées et comparées à celles des autres modèles des processus d'assemblage.

L'objectif du troisième chapitre est l'élaboration d'une méthode systématique d'obtention des graphes de précédence à partir d'un ensemble de gammes d'assemblage. Dans ce but, deux algorithmes sont proposés. Ils sont basés sur la vérification d'une propriété structurelle – la propriété Π – qui est nécessaire et suffisante pour assurer l'équivalence d'un ensemble de gammes à un seul graphe de précédence.

Un état de l'art des méthodes de conception des systèmes d'assemblage issues de l'équilibrage des lignes d'assemblage est présenté dans le quatrième chapitre. Le problème d'équilibrage consiste à trouver l'affectation des tâches aux postes, telle qu'elle assure la minimisation du temps de cycle total. Une approche systémique est proposée par la formulation de ce problème comme problème d'optimisation discrète, en vue de la résolution par la programmation dynamique.

Le dernier chapitre est dédié à l'analyse des systèmes d'assemblage avec auto-équilibrage, dont la conception évite la résolution d'un problème d'équilibrage classique. Il est suffisant qu'un tel système satisfasse une contrainte technologique simple de placement des opérateurs (humains) sur la ligne – du plus lent au plus rapide – pour qu'il atteigne spontanément un comportement optimal du point de vue de l'équilibrage. Une analyse par simulation de tels systèmes est présentée. Ils peuvent être traités comme systèmes dynamiques hybrides à commutations et sauts autonomes. La condition suffisante de l'auto-équilibrage – le "bon ordre" – est démontrée en utilisant les critères de stabilité des systèmes dynamiques discrets.

MOTS CLES

Productique, Assemblage, Graphes de précédence, Équilibrage des lignes,
Systèmes dynamiques hybrides

ABSTRACT

This thesis is a contribution to a global approach of rational design of assembly systems. More precisely, it is dedicated to the generation of precedence graphs to be used in assembly line balancing.

The first chapter of this work contains a survey of the assembly line design.

The second chapter presents a state of the art in the precedence graphs generation methods. The properties of precedence graphs are listed in relation to those of other assembly process models.

The purpose of the third chapter is to develop a systematic method of precedence graphs generation starting from a set of assembly sequences. Two algorithms are proposed, which are based on the existence of a structural property – called property Π - necessary and sufficient to ensure the equivalence between a set of assembly sequences and a single precedence graph.

In the fourth chapter it is presented a state of the art in assembly line balancing methods of design. The assembly line balancing problem consists in the research of that task-to-workstation assignment which ensures the minimisation of the total cycle time. This problem is approached from a systemic point of view, by proposing a resolution based on discrete dynamic programming.

The last chapter is dedicated to the analysis of self-balancing assembly systems, whose design avoids solving an ALB classical problem. It is sufficient that such systems satisfy a simple technological constraint of placing the (human) operators on the line – from slowest to fastest – for spontaneously obtaining an optimal behaviour from the point of view of balancing.
A simulation analysis of such kind of systems is proposed. They can be regarded as hybrid dynamical systems with autonomous switching and autonomous jumps. The sufficient condition for self-balancing – the "well-ordering" – is proved using stability criteria of discrete dynamical systems.

KEYWORDS

CIM, Assembly, Assembly Line Balancing, Precedence Graphs,
Hybrid Dynamical Systems